2004年度　卒業論文

楽譜認識を活用した
演奏支援ソフトウェア

指導　筧捷彦教授

早稲田大学理工学部情報学科

学籍番号:1g01p082-1

板東慶一郎

目 次

第1章　はじめに

1.1　背景

音楽から興味を無くす原因の一つとして，楽譜が読めない事が挙げられる。そこで，学校での音楽教育において読譜指導はどれほど重視されいるものであり，また，学校生徒たちの音楽能力の発達にどのような影響を与えているのかを調べてみた。香西らは子供たちの読譜力や教育現場での読譜指導について調査を行っている [1]。調査によると，教師全体的に読譜指導の意識は低く，その中でも特に小学校教諭が低いことがわかった。また，多くの生徒たちが聴覚記憶に依存して音楽活動に時間を費やしていることも明らかになった。小学校時代からの読譜を重視しない音楽授業の結果，生徒たちは多くを聴覚記憶に依存しながら演奏を行うことを身につけてしまうのだ。また，それが後の読譜力の発達の妨げにもなっているのである。このような現状を打開するためにも，基礎の定着を図らなければならない小学校から中学年までの期間での積極的な読譜指導の導入が重要となるのである。

しかし，週５日制やゆとり教育の導入に伴い音楽の授業時間自体が削減され，読譜指導に割ける時間も少なくなったのも事実である。そのため，授業にも効率的に導入できる読譜指導方法が必要となる。効率的な読譜指導方法の一つとして生徒でも簡単に取り扱えることのできる演奏支援ソフトウェアの利用が挙げられる。

演奏支援ソフトウェアとは，具体的に，楽譜の演奏経過の表示や運指の表示などの視覚的な補助により，演奏の上達，読譜力の向上などを図るソフトウェアのことを指す。演奏支援ソフトウェアは，数多く市販され，また，Web 上でもシェアウェアやフリーウェアのソフトウェアを手に入れることができるようになった。しかし，現状では，ソフトウェアに添付されている曲，専用に市販されている曲，もしくは MIDI ファイとしてで持っている曲に対しての支援しか行ってくれないという点で問題がある。それでは自身が興味を示す楽曲や学校で課題にされた曲の演奏支援ができないため，その役割の果たすところは少ない。特に，自身が興味を示す楽曲は読譜力の発達を促進すという観点に立てば，さまざまな曲に対応した演奏支援ソフトウェアが求められるのである。

1.2　研究の概要

学校での音楽教育における読譜指導の欠落，演奏支援ソフトウェアの対応曲の不十分という問題を解決する方法として，楽譜認識を活用した演奏支援ソフトウェアのプログラム作成を試みた。楽譜認識を活用した演奏支援ソフトウェアとは，自分でスキャンした楽譜の画像を読み込み，認識を行い，演奏支援を行ってくれるソフトウェアである。つまり，自分の必要とする楽曲に対して演奏支援を行ってくれる演奏支援ソフトウェアである。プログラムの構成は大きく分けて楽譜認識部と演奏支援部に分けられる。

楽譜認識部　楽譜認識部ではスキャナで読み込んだ楽譜画像の認識を行う。楽譜上の音楽記号の認識は，それぞれの記号の図形的特徴を基にして行う。今回は小・中学校での音楽教育を対象としたソフトウェアのため，認識の対象となる記号等は小・中学校の音楽の授業で扱うものだけに絞った。音楽記号は，図形的特徴から次の 3 つに分類できる。

- 形を変化させる記号 (音符)…全音符，2 分音符，4 分音符，8 分音符，16 分音符
- 抽象的な図形から成る記号…付点，スタッカート，全休符，2 分休符
- 固有の形状を持つ記号…4 分休符，8 分休符，16 分休符，シャープ，フラット，ナチュラル

形を変化させる記号とは他の記号と接続することで，その形状を変化させるものをさす。音符がこれに当たる。抽象的な図形から成る記号とは，円形や四角形から成る記号をさす。固有の形状を持つ記号は，形を変化させず，抽象的な形状でない記号をさす。それぞれの特徴に見合った認識を行う。楽譜認識から得られた記号の識別，記号や五線の位置情報などからデータを作成し，演奏支援部に渡す。

演奏支援部　演奏支援部では渡されたデータから音程や記号の配置を決定し，演奏支援を行う。演奏支援方法には，MIDI 音源による楽曲の演奏，楽譜の表示，運指の表示の三つがある。MIDI 音源による楽曲の演奏では演奏楽器の選択ができる他，その演奏法や演奏のテンポも選択できる。楽譜を表示では演奏に合わせて楽譜の経過を表示する。運指の表示は小・中学校で主に扱われる楽器であるピアノ (ピアニカ)，リコーダーについて行う。

第2章　楽譜認識の現況

2.1　過去の研究

楽譜認識の研究は1960年代にMIT(Massachusetts Institute of Technology)で始まった。日本では1979年に初めての研究報告があり，それから1980年代には活発的に研究が行なわれるようになった。初期の研究では，単音かつ五線が一段しかないような単純な構成の楽譜を対象としていた。しかし，認識技術の進歩，画像入力装置などの周辺機器を含めたコンピュータの性能の向上により，現在では，和音かつ複数旋律の複雑な楽譜までが楽譜認識の対象となっている。

ニューラルネットワークを認識に用いた楽譜認識の研究も行われている[2]。ニューラルネットワークには，コンピューターでは表現することの難しい人間の感覚的知識や，直感的な知識を活用できるという利点がある。この手法によって，上級者用のピアノ楽譜では93 %以上，初級者用の楽譜に関しては98 %以上の認識率を得ている。

印刷された楽譜だけではなく，手書きの楽譜の認識の研究も行なわれている。最近では，タブレットのペン入力によって音楽記号を入力するオンライン手書き楽譜認識の研究報告もされている[3]。オンライン手書き楽譜認識では，認識方法にDPマッチングとテンプレートマッチングを用いており，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 %の認識率を得ている。しかし，オフラインの手書きの楽譜に関しては，記号表記が様々であることからその認識については課題が残る。

楽譜認識の研究は五線譜だけに留まらず，邦楽譜についても行われている。松島らは尺八譜を対象とした自動認識システムの研究を行なっている[4]。これは尺八の演奏法を示す文字である譜字と拍子線を認識するシステムであり，尺八譜によっては約95 %の認識率を得ている。

2.2 利用実態

1980 年代まで楽譜認識は大学等での研究に限られていた。しかし，1989 年に初めての市販の楽譜認識ソフトウェア「よみとりくん」(日本ソフトバンク) が登場する。その後，ヤマハの「スコアリーダー」，テクノコムの「MIDI スキャン」，河合楽器の「スコアメーカー」などの楽譜認識を用いた市販ソフトウェアが登場する。現在ではスコアメーカーが楽譜認識を用いたソフトウェアの主流となっている。

スコアメーカーは，楽譜認識を用いた楽譜作成ソフトウェアであり，体験版を無料でダウンロードすることができる。スコアメーカーは，手書きの楽譜や手書き風に印刷された楽譜には対応していないものの，五線譜以外にドラム譜やコードネームの認識も可能であり，歌詞も読み取ることができる。その楽譜認識のアルゴリズムは公表されていない。楽譜認識の方法についての比較はできないが，その認識率については実験システムとの比較を行った（５．参照)。

第3章　設計

3.1　画像の二値化

スキャナーから取り込んだ楽譜の画像は完璧な白黒画像ではない。認識の処理をより快適になうため白画素と黒画素の 2 つからなる二値画像に変換する必要がある。

3.1.1　グレイスケール変換

通常スキャナなどから読み込まれるカラー画像では各画素において R,G,B の値で色の情報を確保している。カラー画像を二値画像に変換するためにはいったんグレイスケール (白と黒の濃淡) 画像に変換する必要がある。白と黒の濃淡の割合である輝度レベル Y はカラー画像の R,G,B の値を用いて式 (3.1) のように表すことができる。

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \tag{3.1}$$

この変換を各画素において行う。

3.1.2　閾値

グレイスケールに変換した画像を二値化画像にするめには，求めた輝度に対して，白にするか黒にするかの判断の基準となるような値，つまり，閾値を設定しなければならない。画素の輝度が閾値より低ければ黒画素，高ければ白画素とする。二値化の方法として P タイル法やモード法があるが，これらの方法は人間の判断で閾値を決定する必要がある。判別分析法では濃度ヒストグラムから自動的に閾値を決定することができる。(3.1.3 参照) 閾値の決定には自動的でより適切な値を求めることができる判別分析法を用いた。

3.1.3 判別分析法

判別分析法とは，ある値tを基準として濃度ヒストグラムを2つのクラスに分割したとき，2つのクラス間の分散が大きく，それぞれのクラス内の分散が小さくなるようなtを閾値として定める方法である。つまり，クラス間分散／クラス内分散が最大となるようなtを閾値として定める。

輝度レベルを$C = \{0, 2, ..., 255\}$のように表すとき，輝度レベルがt以下の集合を$C_1\{0, 2, ..., t\}$，tより大きい集合を$C_2\{t+1, ..., 255\}$とする。また，輝度レベルiの画素数を$h(i)$とし，全画素数を$H = h(0) + h(2) + +h(255)$とすると，輝度の正規化ヒストグラム$p(i)(i = 0, 2, \cdots, 255)$は式(3.2)のようになる。

$$p(i) = \frac{h(i)}{H} \tag{3.2}$$

全画素の平均輝度レベルμTは式(3.3)，またその分散は$\sigma^2 T$は式(3.4)のようになる。

$$\mu T = \sum_{i=1}^{255} i * p(i) \tag{3.3}$$

$$\sigma^2 T = \sum_{i=1}^{255} (i - \mu T)^2 p(i) \tag{3.4}$$

2つの累計量$\omega(k)$，$\mu(k)$それぞれを式(3.5)，式(3.6)のようにおく。

$$\omega(k) = \sum_{i=1}^{k} p(i) \tag{3.5}$$

$$\mu(k) = \sum_{i=1}^{k} i * p(i) \tag{3.6}$$

クラスC_1, C_2の生起確率をそれぞれω_1,ω_2とする。

$$\omega_1 = \sum_{i \in C_1} p(i) = \omega(t) \tag{3.7}$$

$$\omega_2 = \sum_{i \in C_2} p(i) = 1 - \omega(t) \tag{3.8}$$

クラスC_1, C_2の平均輝度レベルをそれぞれμ_1,μ_2とする。

$$\mu_1 = \frac{\mu(t)}{\omega(t)} \tag{3.9}$$

$$\mu_2 = \frac{\mu T - \mu(t)}{1 - \omega(t)} \tag{3.10}$$

クラス間分散 $\sigma^2 B$ は式 (3.11) のようになる。

$$\sigma^2 B = \omega_1 \omega_2 (\mu_1 - \mu_2) \tag{3.11}$$

また，クラス間分散とクラス内分散 $\sigma^2 W$ とは式 (3.12) のような関係が成り立つ。

$$\sigma^2 W + \sigma^2 B = \sigma^2 T \tag{3.12}$$

式 (3.12) において，$\sigma^2 T$ が一定より，$\sigma^2 B \to \max$ のとき，$\sigma^2 W \to \min$ となる。つまり，$\sigma^2 B \to \max$ となるとき，$\frac{\sigma^2 B}{\sigma^2 W} \to \max$ となることが分かる。

よって $\sigma^2 B$ を最大とするような t が閾値としてふさわしいといえる。

図 3.1 は全画素の濃度の平均を閾値として二値化した画像である。また，図 3.2 は判別分析法で求めた閾値で二値化した画像である。両者を比較すれば，判別分析法は楽譜を二値化するのに効果的な方法であることがよく分かる。

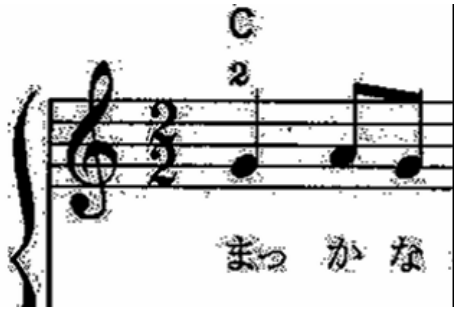

図 3.1:

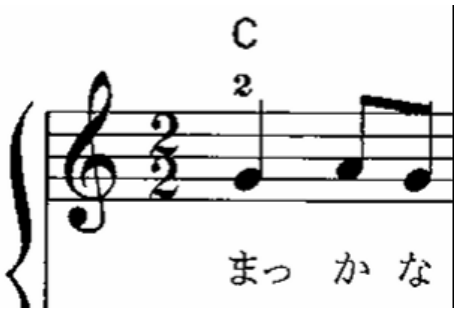

図 3.2:

3.2 楽譜の傾き補正

より正確な認識結果を得るためには，楽譜の印刷の際やスキャナで取り込む際に生じる画像のわずかな傾きを補正する必要がある。傾きの角度は Hough 変換を用いて求める。Hough 変換は線分を抽出する手法の一つである。

3.2.1 Hough 変換

直角座標上で，中心からの距離が ρ で x 座標軸と角度 θ をなす線分と垂直な直線上に点 (x, y) が存在するとしたとき，式 (3.13) の関係が成り立つ。

$$\rho = x\cos\theta + y\sin\theta \tag{3.13}$$

式 (3.13) の関係が成り立つとき，極座標系で (ρ,θ) が分かれば，一つの直線が定まることになる。

基本アルゴリズム　画像中の全ての黒点 (x,y) に対して，式 (3.13) より，θ を 0 度から 1 度づつ加算しながら 180 度まで ρ を計算する。このとき同時に，$\theta - \rho$ 空間と関連付けた 2 次元配列のセルに投票を行う。2 次元配列から投票度数のピークを検索する。投票度数がピークを示す 2 次元配列の θ と ρ をそれぞれ θ_m，ρ_m とするとき，式 (3.14) を示す直線が画像中に存在することが分かる。

$$y = -\frac{\cos\theta_m}{\sin\theta_m}x + \frac{\rho_m}{\sin\theta_m} \tag{3.14}$$

今回，抽出の対象となる線分が五線と決まっているため，x 軸とほぼ水平な線分だけを抽出すればよい。つまり，0 度から 180 度まで ρ を求める必要はなく，傾きの検出する範囲を 80 度から 100 度まで (x 軸に対して $\pm$10 度) とする。また，より正確な傾きの検出を図るために，θ は 0.25 度づつ加算しながら ρ を求め，直線を決定する。

3.2.2 画像の回転

式 (3.14) の θ_m は，x 軸と検出された直線の法線との傾きである。x 軸と直線との傾きは $\theta_m - \frac{\pi}{2}$ となる。$\theta_m - \frac{\pi}{2}$ だけ画像を右回りに回転し，回転した画像を保存する。

3.3　五線の検出・消去

五線　五線は音の高低を示すために使用する。五線を越える音は，加線を使って楽譜に記入する。また，ト音記号などの音部記号が書かれた五線を譜表という。

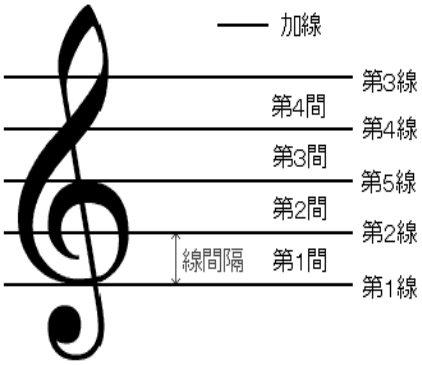

図 3.3: 譜表

3.3.1　検出

傾きを補正した楽譜の画像の水平方向の周辺分布を求めると，例えば図 3.4 のようになる。

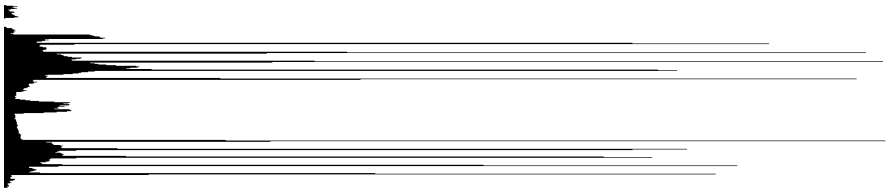

図 3.4: 水平方向の周辺分布

五線を決定するためには，この他に五線の線間隔の情報も必要となる。線と線との間隔の大きさは，白画素のランレングスによって求める。ランレングスとは，白画素または黒画素の連続する長さのことである。二値化した楽譜の画像に対して，縦方向の捜査で白画素のランレングスを調べる。次に，ランレングスの分布を調べ，ピークとなるものを五線の線間隔として決定する。水平方向の周辺分布と五線の線間隔の大きさから五線となる線分を検出し，位置情報も同時に取得する。

小節線も五線と同様に周辺分布から判断して検出し，位置情報を取得する。小節線の検出には図 3.5 のような垂直方向の周辺分布を用いる。

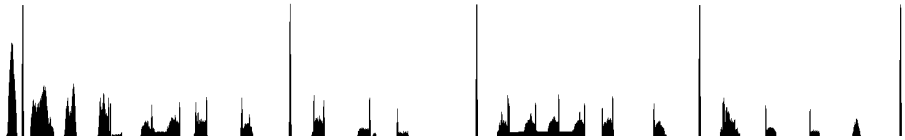

図 3.5: 垂直方向の周辺分布

3.3.2 消去

五線を消去するには，五線の太さを調べる必要がある。二値化した楽譜の画像に対して，縦方向の捜査で黒画素のランレングスを調べる。次に，ランレングスの分布を調べ，ピークとなるものを五線の太さとして決定する。検出した五線の位置から，五線の線の太さ ± 1 に一致する黒点部分を除去する。この手法をとると，記号の一部も消去してしまい，図 3.6 のように記号を分離してしまう。これでは認識が正しく行われない。

図 3.6: 一部が消去された記号

記号の曲線の一部が五線と接触しているとき，記号の一部も五線と判断して消去してしまうことが分かった。また，消去されている範囲は必ず短いことも分かった。記号の一部の消去，分離を避けるために，上の手法において，五線を消去する横の範囲が小さいときは五線を消去しないように設定した。

3.4　ラベリング

ラベリングとは，連結している画素に同じラベル番号を付加することで，複数の領域をグループとして分類する処理である。隣接画素との間の連結性には，4 連結と 8 連結とがある。4 連結とは，上下左右方向の隣接画素で定義される連結性である。8 連結とは，さらに斜めの 4 画素を加えた隣接画素で定義される連結性である。

基本アルゴリズム　ラベリングは次の手順で行う。

1. 画像上を走査，ラベルが付けられていない画素 I を見つけ，新しいラベル番号を付ける。
2. 画素 I について，連結している画素に対し，同じラベル番号を付ける。
3. さらに，今ラベル付けした画素と連結しているすべての画素に同じラベルを付ける。
4. 3 番の操作を，ラベル付けすべき画素が無くなるまで続ける。
5. 4 番の操作が終了したら 1 番に戻り同じ作業を繰り返す。
6. 画像全体について走査をしたときラベリングの処理を終了する。

図 3.7 のように五線を消去した画像に対してラベリングを行い，各音楽記号を切り出しを行う。このとき，ラベリングは譜表ごとに行い，連結領域の捜査は 8 連結に対して行う。また，ラベリングと同時に，ラベル内での x 座標の最大・最小値，y 座標の最大・最小値を取得する。ラベリングの走査は画像の左から行っていくので，画像の左にある記号ほど小さいラベル番号が付けられている。ラベル番号順に，切り出された図形に対して認識を行っていく。

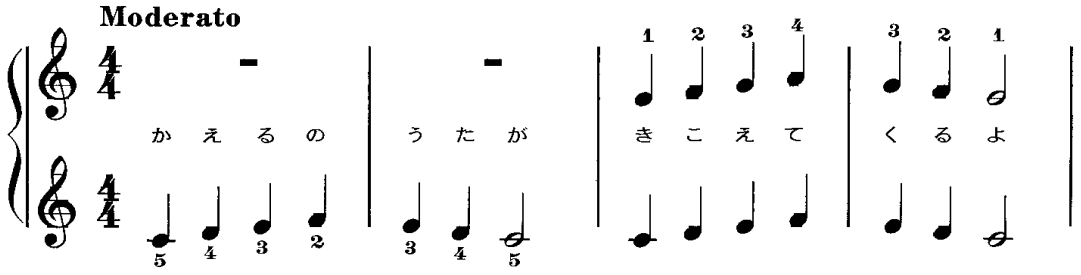

図 3.7: 五線を消去した楽譜

3.5　音部記号の認識

音部記号は, 音名を五線のどの位置に，何の音を基準とするかを示す記号で，譜表の一番左に存在する。音部記号にはト音記号，ヘ音記号，ハ音記号がある。ハ音記号に関しては小・中学校レベルの楽曲ではあまり見られないため認識対象外とした。

ト音記号　ト音記号はソ (ト音) から書き始める記号であり，終点はドを指す。記号は五線全てにまたがる。

図 3.8: ト音記号

ヘ音記号　ヘ音記号はファ(ヘ音) から書き始める記号である。記号は五線の第二線から第五線にまたがる。

図 3.9: ヘ音記号

音部記号の認識は記号の大きさ，譜表の一番左に存在するといった特徴から容易に判断できる。また，譜表の一番左に存在するため，1 番初めに認識される記号である。

3.6　黒符頭を持つ音符の認識

音符は，符頭と呼ばれる丸い玉の部分と，符尾と呼ばれる棒，そして符鉤と呼ばれる旗によって構成される。符頭は音程を示し，黒く塗りつぶされたものを黒符頭とい，中抜きのものを白符頭という。また，符尾の有無，符鉤の数などにより音符の長さが決定される。

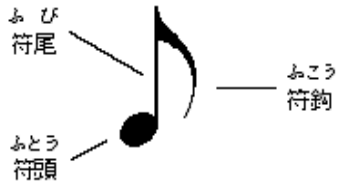

図 3.10: 音符の各部名称

認識を行なう黒符頭を持つ音符は 4 分音符，8 分音符，16 分音符の 3 つである。この他に 32 分音符があるが，小中学校では扱わないので認識は行なわない。

4 分音符　1 小節を $\frac{4}{4}$ 拍子とするとき，1 拍の長さに当たる音符。符鉤を持たない。

図 3.11: 4 分音符

8 分音符　1 小節を $\frac{4}{4}$ 拍子とするとき，$\frac{1}{2}$ 拍の長さに当たる音符。一つの符鉤を持つ。

16 分音符　1 小節を $\frac{4}{4}$ 拍子とするとき，$\frac{1}{4}$ 拍の長さに当たる音符。二つの符鉤を持つ。

図 3.12: 8 分音符

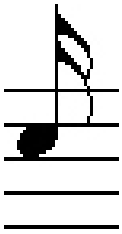

図 3.13: 16 分音符

連桁　8 分音符以下の旗のある音符が続くとき，符尾を連結して書くことができる。このように音符をつなげた表記を連桁という。図 3.14 は，付点 8 分音符，16 分音符，8 分音符の 3 つの音符による連桁である。

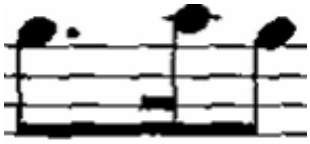

図 3.14: 連桁

3.6.1　黒符頭の検出

五線の間隔を直径とする円を当てはめていき，どれだけ円内に黒点が存在するかを調べる。符頭の中心は必ず，五線・加線上，もしくは五線間・加線間の中心に存在する。このことから，当てはめていく円の中心の y 座標は五線・加線上，もしくは五線間・加線間の中心に固定する。当てはめていく円の中心の x 座標は対象となる図形の横幅の範囲で走査する。黒点の存在の割合が高い円を黒符頭とし，円の中心座標を黒符頭の中心座標とする。

3.6.2 分類

黒符頭を検出した後，黒符頭の示す音符を 4 分音符，8 分音符，16 分音符の中から決定する。

まず，符頭から最も近い符尾を検出する。符尾は一定の長さ以上の垂直線分であるため，垂直方向の周辺分布を基に検出することができる。黒符頭を持つ音符の垂直方向の周辺分布は，図 3.15 のようになる。

図 3.15: 黒符頭を持つ音符の垂直方向の周辺分布

次に，符尾の右，もしくは左に付属する符鉤の数を数え音符を決定する。符鉤の数は，符尾の左右を縦方向に走査したときの連続する黒ランの数から求める。このとき，消去できなかった五線，もしくは加線を，誤って数えてしまうことがある。それを避けるため，連続する黒ランの幅が，五線の太さより大きいことを考慮に入れて，符鉤の数を検出する。

符鉤の数を検出したら，音符の種類を決定する。符鉤が無いときは 4 分音符，1 つのときは 8 分音符，2 つのときは 16 分音符となる。符鉤が符尾のどちらに付属するかは，次のように分類することができる。

- 符尾の右に符鉤がある場合…連桁でない音符，連桁の左端の音符，連桁の中間の音符
- 符尾の左に符鉤がある場合…連桁の右端の音符，連桁の中間の音符

連桁の中間の音符は隣の音符との関係により，符鉤が符尾の右に付属するか左に付属するか決まる。連桁の中間の音符が 16 分音符のとき，右か左どちらかの符鉤が 2 つとなり，もう一方は符鉤が 1 つとなる (図 3.16 参照)。連桁の中間の音符が 8 分音符のとき，右左どちらも符鉤の数は 1 つとなる。これらのことから，符尾の左右の符鉤の数を調べ，数の多い方を優先し，音符を決定する方法をとった。。

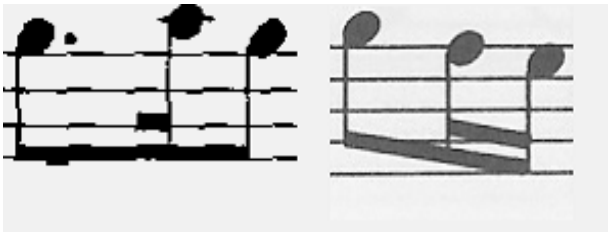

図 3.16: 連桁

3.7　白符頭を持つ音符の認識

白符頭を持つ音符には，全音符，2 分音符がある。

全音符　拍子に関係なく 1 小節全てにあたる長さの音符。符尾，符鉤を持たない。

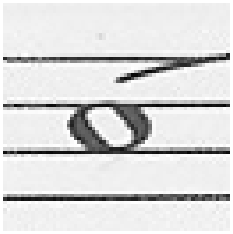

図 3.17: 全音符

2 分音符　1 小節を $\frac{4}{4}$ 拍子とするとき，2 拍の長さに当たる音符。符鉤を持たない。

図 3.18: 2 分音符

3.7.1　白符頭の検出

五線の間隔を直径とする円を当てはめていく。円の外周上に存在する黒画素の割合と，円内の白画素の割合から白符頭であるかどうかを判断する。符頭の中心

は必ず，五線・加線上，もしくは五線間・加線間の中心に存在する。このことから，当てはめていく円の中心の y 座標は五線・加線上，もしくは五線間・加線間の中心に固定する。黒符頭のときと同様に，当てはめていく円の中心の x 座標は対象となる図形の横幅の範囲で走査する。円の外周上に存在する黒画素の割合と，円内の白画素の割合の両方が高い円を白符頭とし，円の中心座標を白符頭の中心とする。

3.7.2 分類

2 分音符の場合，白符頭は符尾の左下，もしくは右上に必ず存在する。これは，和音の場合にも言えることである。また，垂直方向の周辺分布から符尾を検出する。全音符には符尾は無いので，符尾の有無は 2 分音符と全音符とを区別する決定的な条件となる。

全音符は，白符頭のみから成る記号であり，その縦の大きさは五線の線間隔の整数倍とほぼ等しい。これらの条件の他に，記号の大きさなどをもとに，全音符，2 分音符の分類，また，歌詞などの雑音との識別をはかる。

3.8 付点・スタッカートの認識

付点 音符や休符の右隣に存在し，決して五線とは重ならない。左側に存在する音符，または休符の長さを 1.5 倍にする。例えば付点 4 分音符ならば，その長さは 4 分音符＋ 8 分音符と等しくなる。

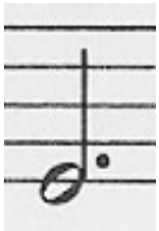

図 3.19: 付点

スタッカート 音符の上もしくは下に存在する。音符の約半分の長さで演奏する。

付点・スタッカートは円形であるという特徴を基に認識を行う。円形であるかどうかの判断には円形度を用いる。

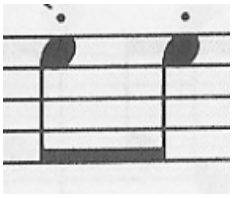

図 3.20: スタッカート

3.8.1 円形度

円形度とは形状の複雑さを測る特徴量である。まず，図形の面積を s，周囲長を l とおく。面積と周囲長は次のように求める。

面積 図形の画素数がそのまま面積の大きさとなる。

周囲長 図形の境界線上の画素数から求める。垂直水平方向を 1 とし，斜め方向の長さは $\sqrt{2}$ とする。

円のときに 1 となるように式 (3.15) で定義する。つまり，e が 1 に近ければ近いほどその形状は円形に近い。

$$e = \frac{4\pi S}{l^2} \tag{3.15}$$

3.8.2 分類

高い円形度を示す図形が得られたならば，その図形が付点であるかスタッカートであるかを識別しなければならない。付点は符頭の右隣，スタッカートは音符の上，もしくは下にあるという性質より，符頭との位置関係から判断できる。

また，円形度による認識を行うと楽譜上の歌詞一部などの雑音まで認識してしまう。これは付点・スタッカートは符頭の近くに存在するということを利用すれば雑音か否かは判断できる。符頭の中心座標と高い円形度を示す図形の中心座標との位置関係から，これらの判断は演奏支援部で行う。

3.9 全休符 2 分休符の認識

全休符と 2 分休符は，その形状が全く同じであり，五線を消去すると長方形が抽出される。

全休符　拍子に関係なく 1 小節全て休む。五線の第三線と第四線の間に存在し，第四線と繋がる。

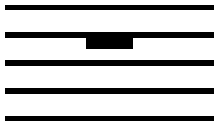

図 3.21: 全休符

2 分休符　1 小節を $\frac{4}{4}$ 拍子とするとき，2 拍の長さだけ休む。五線の第三線と第四線の間に存在し，第三線と繋がる。

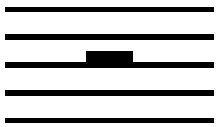

図 3.22: 2 分休符

3.9.1　四角形の認識

図形の面積を S，周囲長を l とおく。このとき，面積 S と周囲長 l を用いて四角形らしさを求める方法を考える。様々な方法が考えられるが，面積 S と周囲長 l の両方を使って四角形らしさを求めることにした。面積 S と周囲長 l が定まっていれば，四角形の縦と横の大きさは一意に求まる。四角形の一辺を x とおく。そのとき，式 (3.16) のような関係が成り立つ。

$$S = x * (\frac{l}{2} - x) \tag{3.16}$$

式 (3.16) の形を変えると，式 (3.17) のような二次方程式を導くことができる。この二次方程式の二つの解がそれぞれ四角形の縦と横の大きさにあたる。

$$2x^2 - lx + 2S = 0 \tag{3.17}$$

判別式 $D = l^2 - 16S > 0$ を満たすものに対し，二次方程式の解と，図形の縦と横それぞれの黒画素の平均値とを比較し認識を行う。

3.9.2 分類

全休符か 2 分休符であるかは，図形の重心の位置で決める。重心は次のように求める。

重心 図形の画素の x,y 座標の平均値から求める。N を全画素数とし，各画素の座標を (x_i, y_i) とすると，重心の座標 (x_g, y_g) は式 (3.18) のようになる。

$$x_g = \frac{1}{N}\sum_{i=0}^{N-1} x_i,\ y_g = \frac{1}{N}\sum_{i=0}^{N-1} y_i \tag{3.18}$$

重心の位置が第四線寄りならば全休符，第三線寄りならば 2 分休符とする。この場合も，付点・スタッカートと同様，雑音を認識してしまうことがあるので，第三線と第四線の間に全休符と 2 分休符が存在するという条件のもとに判断する。

3.10　固有の形状を持つ記号の認識

固有の形状を持つ記号とは，音符のように形を変化することなく，円や四角のように抽象的な形ではない記号を指す。

4 分休符　1 小節を $\frac{4}{4}$ 拍子とするとき，1 拍の長さだけ休む。

図 3.23: 4 分休符

8 分休符　1 小節を $\frac{4}{4}$ 拍子とするとき，$\frac{1}{2}$ 拍の長さだけ休む。

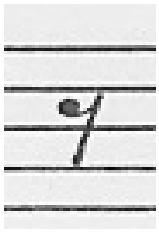

図 3.24: 8 分休符

16 分休符　1 小節を $\frac{4}{4}$ 拍子とするとき，$\frac{1}{4}$ 拍の長さだけ休む。

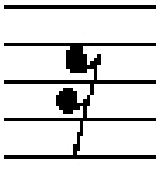

図 3.25: 16 分休符

シャープ　右隣の音符の音程を半音上げる。

図 3.26: シャープ

フラット　右隣の音符の音程を半音下げる。

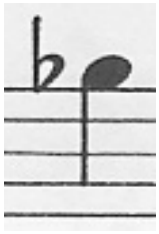

図 3.27: フラット

ナチュラル　半音上がっている，もしくは下がっている音を元に戻す。

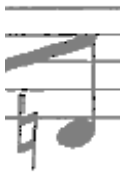

図 3.28: ナチュラル

3.10.1　周辺分布を用いたマッチング

テンプレート画像の縦横の周辺分布と対象となる図形の縦横の周辺分布とを比較し，相互相関によって類似度を求める。これは漢字の文字認識によく用いられる手法である [10]。利点として，比較回数が少ないことが挙げられる。対象となる図形の縦を m，横を n とした場合を例にとる。テンプレートマッチングでは対象となる画像の全画素との類似度の比較が必要なため，比較回数は $m \times n$ 回となる。対して，この手法では，縦横の周辺分布同士を比較するため，比較回数は $m + n$ 回で済むのだ。また，比較回数が少ないことから認識の対象となるものの制限が比較的緩い。

水平・垂直に直線成分を持つものに対しては高い認識率を示すが，持たないものに対しては認識率がやや低いという欠点も持つ。今回の場合，水平・垂直に直線成分を持つものにはシャープ，フラット，ナチュラルがあてはまり，4 分休符，8 分休符，16 分休符はこれに当てはまらない。この欠点については，休符は五線の第一線から第五線の間に必ず存在するという条件を利用し，その条件を満たすものについては認識の基準を下げるということで対処した。

3.10.2　テンプレート

図 3.29 は実際に認識に用いたテンプレート画像とその縦横の周辺分布である。

図 3.29: テンプレート画像と縦横の周辺分布図

3.10.3　相互相関による類似度

2 つの一次関数 $f(x)$，$g(x)$ を考える。この 2 関数のユークリッド距離 d は式 (3.19) のように定義される。

$$d = \sqrt{\int_a^b (f(x) - g(x))^2 dx} \tag{3.19}$$

また，ユークリッド距離 d から式 (3.20) を導くことができる。

$$d^2 = \int f^2(x)dx + \int g^2(x)dx - 2\int f(x)g(x)dx \tag{3.20}$$

式 (3.20) の $\int f(x)g(x)dx$ は関数間の内積を表している。正規化すると式 (3.21) のようになる。これを相互相関という。

$$\cos\theta = \frac{\int_a^b f(x)g(x)dx}{\sqrt{\int_a^b f^2(x)dx}\sqrt{\int_a^b g^2(x)dx}} \tag{3.21}$$

式 (3.21) において，θ は関数間の角度であり，2 つの関数が同じ周期の周期関数であれば関数間の位相差を示す。これより，$\cos\theta$ の値が大きい，つまり 1 に近いほど，2 つの関数が類似していることを表し，この値を類似度という。

3.11 データの作成

演奏支援部へ送るデータは次のようになっている。

1. 画像の縦・横の大きさ
2. 五線の線間隔
3. 五線の位置情報 (y 座標)
4. 小節線の位置情報 (x 座標)
5. 記号情報 (ラベル番号，中心の x 座標，中心 y 座標，記号の識別)
   ただし，音符のときは符頭の中心座標を表記する

演奏支援部では五線の位置情報と符頭の中心座標から音程を決定する。ラベル番号は，和音もしくは連桁を判断するのに用いる。

第4章　実装

4.1　楽譜認識

スキャナで読み込んだ画像は，図 4.1 のような手順で楽譜認識を行う。

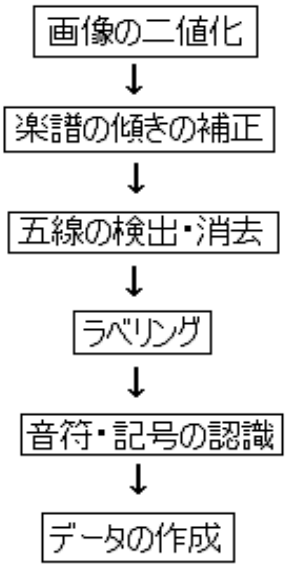

図 4.1: 楽譜認識

1. 画像の二値化　スキャンされた楽譜の画像に対し、判別分析法を用いて閾値を求めの二値化を行う。

2. 楽譜の傾き補正　ハフ変換によって楽譜の傾きの角度を求め、傾きの補正を行う。

3. 五線の検出消去　横方向にプロジェクションした周辺分布から五線を決定する。五線の位置情報を取得したら五線を消去する。

4. ラベリング　五線が消去された画像には記号だけが残る。これらの記号にラベリングを行い，記号を切り出す。

5. 音楽記号の認識　切り出された記号について、記号の形状的な特徴をもとに認識を行う。

6. データの作成　記号の認識によって得られた記号の識別、記号や五線の位置情報などのデータを作成、演奏支援部に渡す。

4.2　音符・記号の認識

音部記号が認識されたら，図 4.2 のような手順で音符・記号の認識を行う。記号の出現頻度や，認識の正確性から図 4.2 のような順番で認識を行うことにした。

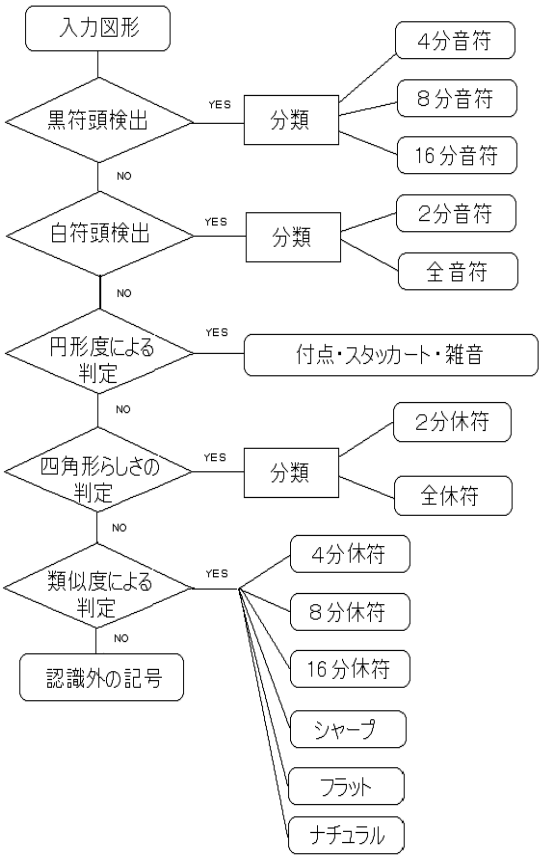

図 4.2: 音符・記号の認識

4.3 楽譜データファイル

演奏支援部に渡すデータファイル，GKF ファイルについて説明する。記号の識別番号は次のように設定した。

番号	記号	番号	記号	番号	記号
-1	全休符	1	全音符	6	付点、スタッカート、雑音
-2	2分休符	2	2分音符	7	ト音記号
-3	4分休符	3	4分音符	8	ヘ音記号
-4	8分休符	4	8分音符	9	シャープ
-5	16分休符	5	16分音符	10	フラット
				11	ナチュラル

図 4.3: 記号の識別番号

次に示すのは，実際，認識によって得られた GKF ファイルの一部である。

h=2338
w=1700
i=15
s=1181,1198,1214,1229,1245
b=179,601,964,1322,1653,1664
2,200,1219,8
3,221,1220,6
4,221,1236,6
8,271,1206,4
8,309,1237,4
8,352,1221,4
8,398,1237,4

”h=”,”w=”の後の数値はそれぞれ画像の縦の大きさ，横の大きさである。その次の，”i=”の後の数値は五線の線間隔で，”s=”の後の 5 つの数値は，各五線の位置情報 (y 座標) を示す。また，”b=”の後の 6 つの数値は各小節線の位置情報 (x 座標) を示す。それから下の列の数値は，記号情報である。左から，ラベル番号，中心の x 座標，中心 y 座標，記号の識別の順番に並んでいる。下から 4 列の記号情報は，全て同じラベル番号である。記号の識別番号 4 は，8 分音符を示す。このことより，ラベル番号 8 の示す記号は，4 つの 8 分音符から成る連衡であることが分かる。

第5章　実験評価

5.1　方法と結果

楽譜認識を活用した演奏支援ソフトウェアの楽譜認識部である楽譜認識システムを用いて，実際に楽譜認識を実行した。また，河合楽器のスコアメーカーについてもも同じ楽譜に対して，楽譜認識を実行し，楽譜認識システムとの認識率の比較を行った。認識の対象となる楽譜には，小中学校レベルの代表的な楽曲を用いた。認識の対象に用いた楽譜は次の 10 曲である。

- 春が来た
- さくらさくら
- ぶんぶんぶん
- チュウリップ
- こいのぼり
- カエルの合唱
- キラキラ星
- 赤鼻のトナカイ
- きよしこの夜
- あわてんぼうのサンタクロース

認識率は音符と休符も含む記号に分けて求めた。このような分け方をしたのは，音符は形を変化させる音楽記号であり，スコアメーカーでも認識方法その他の音楽記号と違うであろうと推測されるからだ。付点の付いた音符に関しては音符と付点の両方が認識されて認識されたものとした。

歌詞やコードなどの文字を音楽記号として認識してしまうことがいくつか見られた。そのような場合には，過多認識としてカウントした。

楽曲	音符		記号		合計		過多認識
	認識数	認識率	認識数	認識率	認識数	認識率	
春が来た	91/91	100	7/7	100	98/98	100	0
さくらさくら	104/107	97.2	23/24	95.8	127/131	96.9	2
ぶんぶんぶん	59/60	98.4	6/6	100	65/66	98.4	0
チューリップ	74/74	100	7/7	100	81/81	100	0
こいのぼり	99/99	100	13/13	100	112/112	100	0
カエルの合唱	40/40	100	14/14	100	54/54	100	0
キラキラ星	137/137	100	8/8	100	145/145	100	0
赤鼻のトナカイ	90/93	96.8	13/13	100	103/106	97.1	0
きよしこの夜	75/75	100	6/6	100	81/81	100	3
あわてんぼうのサンタ	135/136	99.2	19/20	95	154/156	98.7	0
合計	904/912	99.1	116/118	98.3	1020/1030	99.0	5

表 5.1: 楽譜認識システムによる認識結果

楽曲	音符		記号		合計		過多認識
	認識数	認識率	認識数	認識率	認識数	認識率	
春が来た	91/91	100	7/7	100	98/98	100	0
さくらさくら	106/107	99.1	24/24	100	130/131	99.2	0
ぶんぶんぶん	60/60	100	6/6	100	66/66	100	0
チューリップ	74/74	100	7/7	100	81/81	100	0
こいのぼり	99/99	100	13/13	100	112/112	100	0
カエルの合唱	40/40	100	14/14	100	54/54	100	0
キラキラ星	137/137	100	8/8	100	145/145	100	0
赤鼻のトナカイ	93/93	100	13/13	100	106/106	100	0
きよしこの夜	72/75	97.3	6/6	100	78/81	97.5	1
あわてんぼうのサンタ	136/136	100	19/20	95	155/156	99.3	1
合計	909/912	99.7	117/118	99.2	1026/1030	99.6	2

表 5.2: スコアメーカーによる認識結果

5.2 考察

実行の結果，表 5.1，表 5.2 のような認識結果が得られた。全 10 曲，約 1000 個の記号に対して，楽譜認識システムでは 99.0 %の認識率を，スコアメーカーでは 99.6 %の認識率を得た。全体の認識率では 0.6 %，認識記号数では 6 個，スコアメーカーが上回る結果となった。また，音符，記号ともにスコアメーカーのほうが高い認識率を示した。その他，出現する記号数が多いほど，楽譜認識システムの認識率は低い傾向にあることも分かった。しかし，抽象的な形状の記号，黒符頭を持つ音符の認識に関しては，スコアメーカーと同等，もしくはそれ以上の認識精度を示した。

5.2.1 誤認識された記号

この実験の中で，過多認識，認識違い，認識されないなどの，誤認識された記号について調べてみた。図 5.1，5.2 はそれぞれ楽譜認識システムが誤認識した記号，スコアメーカーが誤認識した記号をグラフに表したものである。

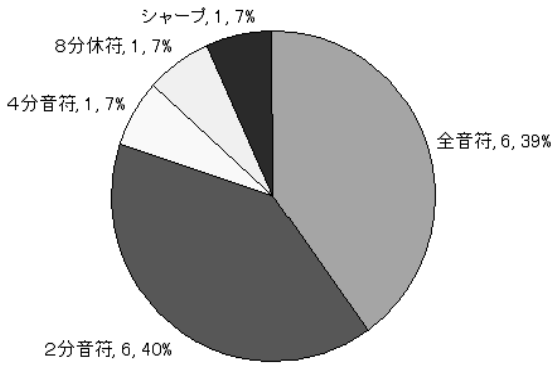

図 5.1: 楽譜認識システムが誤認識した記号

楽譜認識システムで誤認識を招いた原因には，次のような事が考えられる。

- 五線の消去が適切に行われなかった
- 認識基準が適切でない
- 楽譜の印字ミス

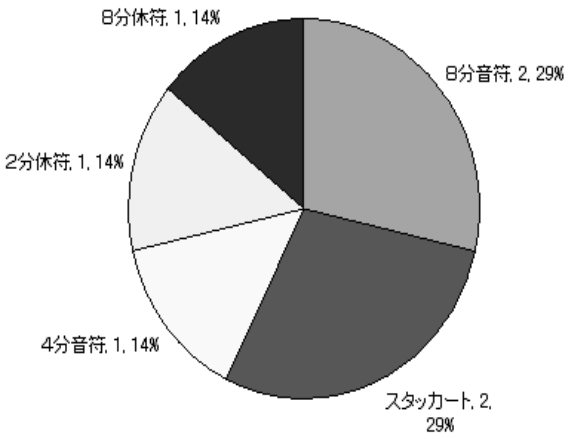

図 5.2: スコアメーカーが誤認識した記号

シャープの誤認識は五線の消去によるものである。五線が消去されないために，シャープが黒符頭と接続てしまい，認識されなかったのだ。また，8 分休符の場合にも同じ理由である。

一番誤認識されている記号は全音符，2 分音符である。これらの記号の誤認識の原因は認識の基準である。全音符は，歌詞やコードを間違って認識してしまっている。過多認識は全て全音符である。2 分音符は，認識されない場合が多かった。

高い認識を示していた黒符頭を持つ音符が唯一認識できない場合があった。楽譜の印字ミスによって，一部途切れてしまった 4 分音符である。この音符はスコアメーカーでも認識されなかった。

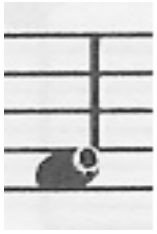

図 5.3: 楽譜の印字ミス

5.2.2　二値化精度の比較

スコアメーカーで楽譜認識を行っているとき，五線が検出できないとして，認識に至らない楽譜があった。スコアメーカーでは画像をいったん二値に変換した画像の保存し，認識を行う。図 5.4 はスコアメーカーでによって二値化された画像であるが，閾値が適切に設定されていないためか，スキャンの際の影が黒画素として出てしまっている。これが弊害となり五線が検出なかったものと考えられる。これに対して，図 5.5 は楽譜認識システムで二値化した画像である。適切に二値化が行われていることが分かる。このため，認識も正常に行われた。

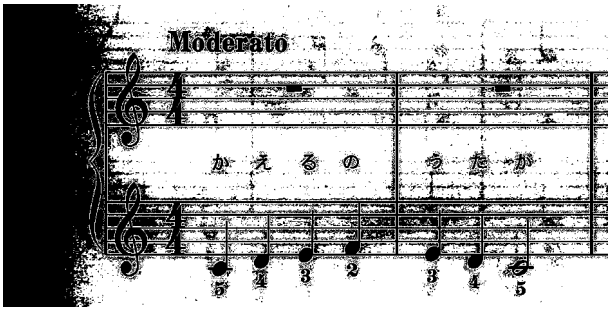

図 5.4: スコアメーカーによる「かえるの歌」の二値化画像

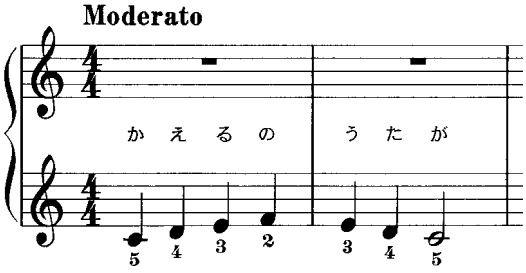

図 5.5: 楽譜認識システムによる「かえるの歌」の二値化画像

5.2.3　画像の傾きへの対応の比較

楽譜のスキャンの際に生じる傾きへの対応を比較した。図 5.6 は約 3 度の傾きをつけてスキャンした画像である。この極端な傾きのついた画像に対して，楽譜認識が正常に行われるかどうかを調べた。

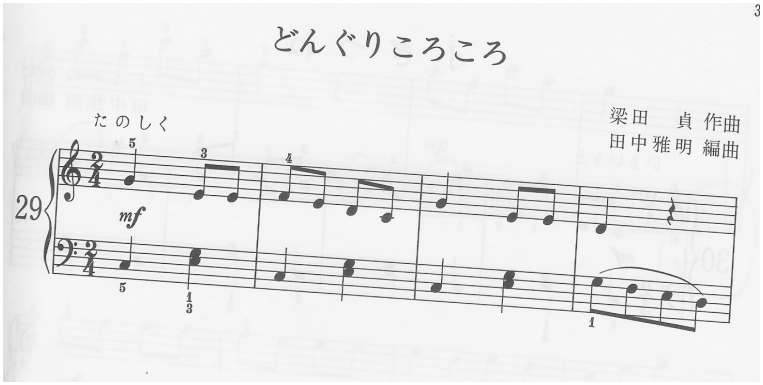

図 5.6: 極端な傾きのついた画像

表 5.3 からも分かるように，極端な傾きによって認識率が低くなるといったことは楽譜認識システム，スコアメーカーともに見られなかった。スコアメーカーがどのような手法で，画像の傾きに対応しているのかは不明である。しかし，楽譜認識システムにおいて，Hough 変換を用いた楽譜の傾き補正が，適切に行われていることが分かった。

	音符		記号		合計	
	認識数	認識率	認識数	認識率	認識数	認識率
スコアメーカー	24/24	100	3/3	100	27/27	100
楽譜認識システム	24/24	100	3/3	100	27/27	100

表 5.3: 極端な傾きのついた画像の認識結果

第6章　研究のまとめ

6.1　今後の課題

実験において，抽象的な形状の記号，黒符頭を持つ音符の認識，画像の二値化については，市販の楽譜認識ソフトウェアに劣らぬ結果を示した。しかし，さまざまな問題も浮き彫りとなった。楽譜認識を活用した演奏支援ソフトウェアの楽譜認識部には次のような課題がまだ残る。

- 五線の消去法，白符頭検出法の改善
- 印字ミスにも対応できる認識法の必要性
- より複雑な楽譜への対応
- 認識対象となる音楽記号の種類の増加
- 認識の高速化

実験の結果より，五線の消去が不適切であったり，白符頭が適切に検出されない事により，記号を誤認識してしまう場合が多いことが分かった。記号の一部を消去することなく，また，記号同士を接続しないような五線消去法が求められる。白符頭の検出法に関しても改善の必要がある。

今回，実験で認識を行った楽曲はすべて初級者向けの楽譜である。楽譜認識を活用した演奏支援ソフトウェアの活用の幅を広げる意味でも，より複雑な楽譜への対応が求められる。また，そのために，認識対象となる音楽記号の種類を，タイ，スラー，速度記号などさらに追加することも必要である。

考察では述べなかったが，認識の速度についても遅いという問題がある。スコアメーカーでは，100 個程度の記号から成る楽譜に対し，二値化から認識結果表示までの時間はおよそ 10 秒である。楽譜認識システムでは，二値化から楽譜データ作成までの時間は 15 秒程度であり，そこから演奏支援部にデータを渡し，音程などを決定する作業を行うと，その時間はさらにかかってしまう。さまざまなアプローチによる認識の高速化も必要となる。

6.2　楽譜認識の今後

音楽を専門の職業とする人を対象とした楽譜認識ソフトウェアの開発を考えると，より正確な楽譜認識が求められる。1 ページの楽譜の認識において，記号の認識率が 1 %でも低いと，およそ 1〜2 個の記号が認識できなかった事を意味する。1 つの音楽記号が抜けている，もしくは誤っていると，曲は違った物になってしまう。1 %の認識率の向上が非常に重要になってくる。

また，オフラインでの手書きの楽譜認識の技術のさらなる向上が求められる。オフラインでの手書きの楽譜認識の研究を掘り下げていくことで，印字楽譜の認識の技術の向上にも必ず繋がるであろう。

謝辞

この研究をするにあたり，様々な御指導をしてくださった筧捷彦教授，長慎也氏，音楽について御指導してくださった秘書の磯久さん，一緒に研究に取り組んだ鈴木秀一君，研究室の同僚の皆さんにこの場を借りて感謝の意を表します。

関連図書

[1] 「子どもの読譜力の発達に関する研究」　香西久美子
http://www.art.hyogo-u.ac.jp/hrsuzuki/students/peco99.pdf

[2] 「ニューラルネットワークを用いた印刷楽譜の認識」　宮尾秀俊　1997 年
http://sunak2.cs.shinshu-u.ac.jp/ miyao/Doctor/doctor.html

[3] 「オンライン手書き楽譜認識」　宮尾秀俊, 中村隆

[4] bit 別冊「コンピュータと音楽の世界」共立出版

[5] 「C 言語による画像処理入門」 安居院猛，長尾智晴　昭晃堂　2000 年

[6] 「コンピュータ画像処理」 安居院猛，中嶋正之　秋葉出版　1989 年

[7] 「ディジタル画像処理入門」　酒井幸市　コロナ出版社　1997 年

[8] 「Visual Basic & Visual C++ によるデジタル画像処理入門」　酒井幸市
CQ 出版社　2002 年

[9] 「統計的画像処理手法」　栗田多喜夫
http://www.neurosci.aist.go.jp/ kurita/lecture/statimage/statimage.html

[10] 「文字の構造情報を活用した印刷文字認識方式」　木村正行　馬場伊美子

付録

BMP クラス　ビットマップイメージをに関する処理を行う

void turnImage(char* fname,double angle);
//画像を angle だけ右回りに回転する。

BOOL binalizeBMP(char* fname);
//画像を二値化する。成功すれば TRUE を返す

Line クラス　五線や小節線に関する処理を行う

int runLength(int** image,int height,int width,int bow);
//bow=1 ならば黒，bow=0 ならば白画素のランレングスを調べ，その値を返す。

int getStaves(int** image,int height,int width,int wlng);
//五線を検出，位置情報を取得する。五線の本数を返す。

int getStaveInterval();
//五線の線間隔を取得，その値を返す。

void removeStaves(int** image,int height,int width,int blng,int wlng);
//五線を消去する。

double getBar(int** image,int height,int width);
//小節線を検出，位置情報を取得する。小節線の平均本数を返す。

void writeStavesData(FILE *data,int num);
//五線・小節線の位置情報をデータに書き込む。

double getAngle(int** image,int height,int width);
//Hough 変換により，楽譜画像の傾きの角度を検出し，その値を返す。

Symbol クラス　音楽記号の認識を行う

int labeling(int** image,int h1,int h2,int w1,int w2);
//ラベリングを行う。総ラベル数を返す。

void getSymbols(int** image,int h1,int h2,int w1,int w2,int st[],FILE *data);
//記号の認識を行う。

BOOL getClef(int** image,int l,FILE *data);
//音部記号の認識を行う。認識すれば TRUE を返す。

BOOL getBlackHead(int** image,int l,double slide,FILE *data);
//黒符頭の検出を行う。検出すれば TRUE を返す。

int classifyBlackHead(int** image,int l,int x,int y,int i,int num);
//黒符頭の音符を分類する。を返す。

BOOL getWhiteHead(int** image,int l,double slide,FILE *data);
//白符頭の検出，分類を行う。認識すれば TRUE を返す。

BOOL getDotOrRest(int** image,int l,double slide,FILE *data);
//円形の記号，もしくは四角形の記号の認識を行う。どちらか認識すれば TRUE を返す。

BOOL getOtherSymbols(int** image,int l,double slide,FILE *data);
//固有の形状を持つ記号の認識を行う。認識すれば TRUE を返す。