

2019 年度 卒業論文

コーヒー抽出に関する音声認識可能な Web レシピの開発

指導教員 須田 宇宙 准教授

千葉工業大学 情報ネットワーク学科

須田研究室

1632130 氏名 肥田雄也

提出日 2020 年 1 月 25 日

目次

第 1 章 緒言	1
第 2 章 コーヒーについて	2
2.1 コーヒーとは	2
2.2 コーヒー(アラビカ種)の起源と伝搬の歴史	2
2.3 コーヒーの消費量の推移	3
第 3 章 抽出器具について	4
3.1 抽出器具	4
3.1.1 プアオーバー	4
3.1.2 コーヒープレス	5
3.1.3 エスプレッソ	5
3.1.4 ソロフィルター	5
3.2 抽出方式	6
3.2.1 浸漬式	6
3.2.2 透過式	7
3.2.3 高圧抽出式	7
3.3 その他の記号	9
3.3.1 小節	9
3.3.2 休符	10
3.3.3 拍子	11
3.3.4 テンポ	11
3.3.5 調号	12
第 4 章 ピアノの指導方法	13
4.1 ピアノの基本的な指導方法	13
4.2 個人ピアノ教室の実態	13
第 5 章 画像認識ツール	14
5.1 画像認識とは	14
5.2 OpenCV	14
第 6 章 Python について	15
6.1 Python とは	15
6.2 日本語を出力する Python ライブラリについて	15
第 7 章 本研究で開発するシステムの概要	16
7.1 実装機能	16

7.2	楽譜ならではの画像認識について	17
7.3	楽譜認識	17
7.3.1	音部記号の認識	18
7.3.2	五線の検出	19
7.3.3	音符の検出	21
7.4	音階出力	23
7.4.1	五線の分割	23
7.4.2	五線の中央の線の検出	23
7.4.3	1 音階の範囲設定	24
7.4.4	音階を求める	24
7.4.5	音階の切り替え	25
7.4.6	音階を表示する	25
第 8 章 システムの有用性の検証		26
8.1	検証方法	26
8.2	調査方法	26
8.3	アンケート結果	27
第 9 章 結言		32
第 10 章 参考文献		34
付録 A 作成したプログラム		35

図目次

2.2-1	UCC ホームページより:コーヒーの軌跡	2
2.3-1	楽器演奏人口の上位 5 都府県	3
3.1-1	五線譜の一例	4
3.1-2	終止符	4
3.2-1	ト音記号	6
3.2-2	ヘ音記号	6
3.2-3	ハ音記号	6
3.2-4	1 オクターブ	7
3.2-5	音符	7
3.2-6	全音符	8
3.2-7	2 部音符	8
3.2-8	4 分音符	8
3.2-9	8 分音符	9
3.2-10	16 部音符	9
3.3-1	小節	9
3.3-2	全休符	10
3.3-3	長休符	10
3.3-4	拍子	11
3.3-5	調号	12
3.3-6	シャープ記号	12
3.3-7	フラット記号	12
3.3-8	シャープ調号の変化	12
3.3-9	フラット調号の変化	12
5.2-1	顔認識	14
6.2-1	Pillow を用いて加工を行った画像	15
7.1-1	本システムのフローチャート	16
7.3-1	機械学習	19
7.3-2	ト音記号の認識	20
7.3-3	片面スキャンした楽譜	20
7.3-4	成功した五線検出結果	21
7.3-5	重複した音符検出結果	21
7.3-6	検出を絞った出力結果	22
7.3-7	音符の中央の検出結果	22
7.4-1	五線を 5 本ずつに分ける	23
7.4-2	五線の中央	23
7.4-3	1 音階の範囲	24
7.4-4	音階の求め方	24

7.4-5	出力画像	25
8.3-1	音階を手書きで対処するかの設問結果	27
8.3-2	システムの必要性の設問結果	29
8.3-3	時間的有用性の設問結果	30

表目次

3.3-1	その他の休符	10
3.3-2	テンポ表記	11
8.2-1	設問内容	26
8.3-1	問2の回答結果	28
8.3-2	問5の回答結果	31

第1章 緒言

「悪魔のように黒く、地獄のように熱く、天使のように純粋で、愛のように甘美である。」これはフランスの外交官である、シャルル＝モーリス・ド・タレーラン＝ペリゴールが遺したコーヒーの名言である。コーヒーはただの飲料物でありながらも、長い年月をかけて愛され、世界の人々を魅了してきた。

それは日本も例外ではない。江戸時代初期に長崎の出島に持ち込まれた際は、一部の人間しか飲用はできなかつたが、明治時代に文明開化が起きるとみるみるうちに一般層に普及していった。現代では、国際機関コーヒー機関（ICO）の「世界の国別コーヒー消費量」で4位を記録しているほどである。

私はこの度の研究に際して、コーヒーをテーマにした研究に興味を持った。

しかし、初心者にとってピアノの楽譜は難易度が高く、楽譜が読めないことで挫折する人が多い。

そこで、ピアノ教室では各音符に手書きで音階を書き込む工夫がされているが、指導者にとって時間的コストがかかっていることが問題点としてあげられる。その問題を解決するために、本研究では実際に音階付加システムを開発し、ピアノ指導者に評価してもらうことを目的とする。

第2章 コーヒーについて

本章では、コーヒーの歴史や流行について説明する。

2.1 コーヒーとは

コーヒーとは、コーヒーノキという樹木から採取される種子を焙煎し、お湯等で成分を抽出した飲料物である。主に北回帰線と南回帰線を挟むコーヒーベルトと呼ばれる地域で栽培されており、数百種類の品種が存在する。商業生産としては、高品質であるが栽培の難しいアラビカ種がおよそ 60 % 前後、品質はアラビカ種に劣るが耐病性に優れ、大量生産向きであり缶コーヒーなどに用いられるロブスタ種がおよそ 40 % 前後を占める。抽出方法についても日本の純喫茶でよく見られるプアオーバーや、サイフォンの他に、フレンチプレス、エスプレッソ、ソロフィルターなど、その目的や表現したい風味に従い、多くの数が存在し使い分けられている。

2.2 コーヒー(アラビカ種)の起源と伝搬の歴史

コーヒー(アラビカ種)はエチオピアのアビシニア高原にて発見された。その後、アラビアに伝播しオランダの貿易商人達の手によってアフリカやアジアへと広がっていき、商業用生産が活発になる。アフリカやアジア各国にコーヒーが広がり、フランスに至ると、とある海軍兵士がコーヒーの苗木を当時フランス領であったマルティニーク島に持ち出すことになる。こうしてラテンアメリカにもコーヒーは広がり、温暖でコーヒーの栽培に特に適した北回帰線と南回帰線を挟むコーヒーベルトにおいて栽培は進んでいった。現代でも、コーヒーベルト各国はコーヒーシェアのほとんどを占めている。グアテマラ・ブラジル・コロンビア・スマトラなどは、コーヒーを普段飲まない人でも聞き馴染みのある生産地であろう。

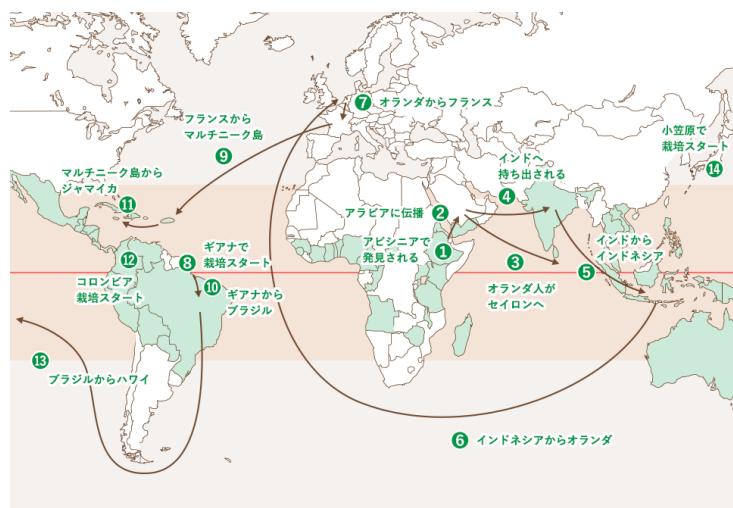


図 2.2-1: UCC ホームページより:コーヒーの軌跡

2.3 コーヒーの消費量の推移

日本の楽器人口は 500 万～600 万人ほどいるといわれている。それはつまり、全国民の約 5 % が何かしらの楽器を演奏できるということになる。総務省統計局が実施した社会生活基本調査に、この 1 年間に楽器を演奏した 25 歳以上の人口があった。25 歳未満は学生が多く、大人と行動パターンが異なる可能性があるため除外して統計をとっている。全国の 25 歳以上楽器演奏人口は 809 万 3000 人で、25 歳以上人口 100 人あたり 8.18 人であった。楽器演奏人口が最も多いのは東京都で、25 歳以上人口 100 人あたり 12.07 人 (偏差値 84.1) であった。続いて 2 位は神奈川県で 10.27 人、3 位以下は滋賀県 (9.97 人)、京都府 (9.33 人)、兵庫県 (9.17 人) の順で都市部が上位に多い。グラフを図 2.3-1 に示す。一方、最も楽器演奏人口が少ないのは長崎県で 25 歳以上人口 100 人あたり 4.75 人 (偏差値 32.8) であった。これに青森県 (4.82 人)、高知県 (5.35 人)、山梨県 (5.40 人)、福島県 (5.50 人) と続いている。

14

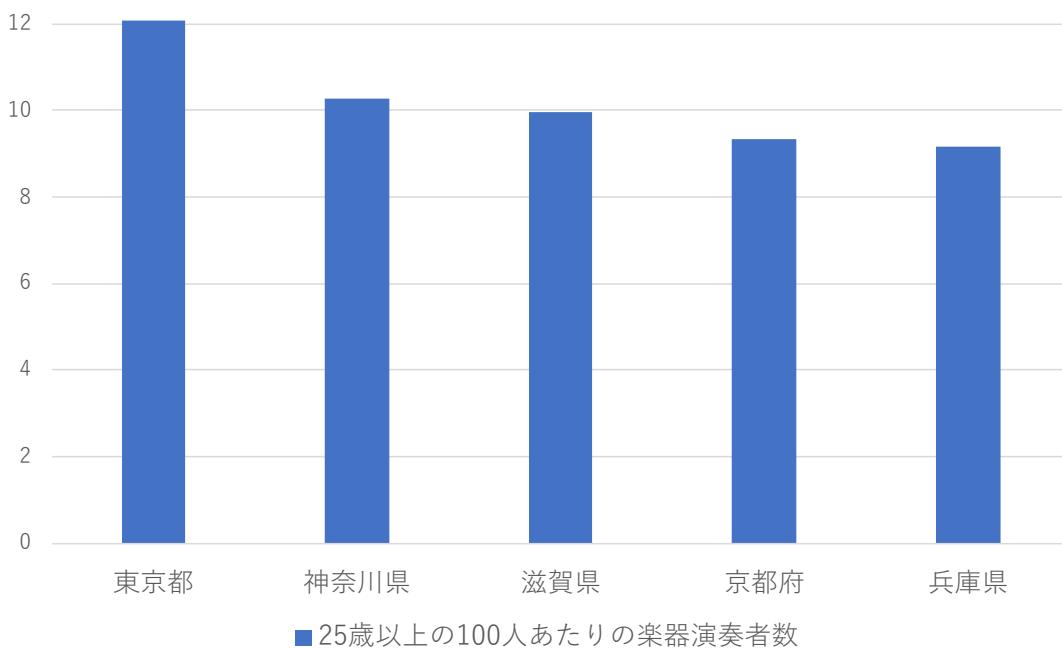


図 2.3-1: 楽器演奏人口の上位 5 都府県

第3章 抽出器具について

本章では、楽譜の基本的な記号等を説明する。

3.1 抽出器具

そもそも楽譜とは、楽曲を演奏記号や符号などの記号によって書き記されているものである。現在もピアノやバイオリンなどの弦楽器をはじめ、金管楽器や木管楽器など様々な楽器の楽譜が世界に存在している。

5本で1組となる平行な直線を五線と言い、五線は相対的な音の高さを表している。そして、五線が数段描かれているものを五線紙といい、そこに音符という音を表す記号を書き込んだものが五線譜である。五線譜の一例を図3.1-1に表す。



図3.1-1: 五線譜の一例

五線譜では、音符の位置が上になるほど音階は高音域になり、下の位置になるほど低音域になる。楽曲が進むにつれて左から右に進んでいき、図3.1-2の終止線をもって終了することになる。

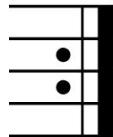


図3.1-2: 終止符

3.1.1 プアオーバー

五線譜の各行の左端、音部記号と呼ばれる記号を付与する。ヴァイオリン記号・ソプラノ記号・アルト記号・テノール記号・バス記号等いくつかの種類が存在する。その中でも一般的に使われる的是ヴァイオリン記号とバス記号であり、それぞれト音記号とヘ音記号と呼ばれる。また、ト音記号とヘ音記号の他にハ音記号と呼ばれる記号が存在している。ソプラノ記号等はハ音記号で表されている。

3.1.2 コーヒープレス

五線譜の各行の左端，音部記号と呼ばれる記号を付与する。ヴァイオリン記号・ソプラノ記号・アルト記号・テノール記号・バス記号等いくつかの種類が存在する。その中でも一般的に使われるのはヴァイオリン記号とバス記号であり，それぞれト音記号とヘ音記号と呼ばれる。また，ト音記号とヘ音記号の他にハ音記号と呼ばれる記号が存在している。ソプラノ記号等はハ音記号で表されている。

3.1.3 エスプレッソ

五線譜の各行の左端，音部記号と呼ばれる記号を付与する。ヴァイオリン記号・ソプラノ記号・アルト記号・テノール記号・バス記号等いくつかの種類が存在する。その中でも一般的に使われるのはヴァイオリン記号とバス記号であり，それぞれト音記号とヘ音記号と呼ばれる。また，ト音記号とヘ音記号の他にハ音記号と呼ばれる記号が存在している。ソプラノ記号等はハ音記号で表されている。

3.1.4 ソロフィルター

五線譜の各行の左端，音部記号と呼ばれる記号を付与する。ヴァイオリン記号・ソプラノ記号・アルト記号・テノール記号・バス記号等いくつかの種類が存在する。その中でも一般的に使われるのはヴァイオリン記号とバス記号であり，それぞれト音記号とヘ音記号と呼ばれる。また，ト音記号とヘ音記号の他にハ音記号と呼ばれる記号が存在している。ソプラノ記号等はハ音記号で表されている。

3.2 抽出方式

ここでは、楽譜を読む上で必須である知識を説明する。

3.2.1 浸漬式

音部記号の中で汎用性が高いものはト音記号とヘ音記号である。ト音記号とは図 3.2-1 の形をしており、高音域を表している記号である。ピアノでは多くの場合右手で演奏する。名前の由来は書き始める場所から来ていて、ト音記号は音階のソの位置から書き始める。ソは日本語表記の音階でトにあたるため、ト音記号という名称になっているのである。



図 3.2-1: ト音記号

ヘ音記号は図 3.2-2 の形をしており、低音域を表している記号である。ピアノでは多くの場合左手で演奏する。名前の由来はト音記号と同様、書き始めの場所がファであることから、日本語表記の音階でヘにあたるため、ヘ音記号という名称になっている。



図 3.2-2: ヘ音記号

そして、ト音記号とヘ音記号の他にハ音記号という記号がある。形は図 3.2-3 の形をしており、中音域を表すために使われる。別名中音部記号と言い、古典派以前はソプラノ・アルト・テノールなどの声域の表記の為に使われていた。表す音域はト音記号とヘ音記号の中間で、ハ音記号のみで広い音階に対応できる。



図 3.2-3: ハ音記号

3.2.2 透過式

音階とは音楽において用いられる音を、高さ順に配列したものである。音階は1オクターブを1周期として一定の音程関係で表示される。1オクターブには12個の音が存在しており、鍵盤では図3.2-4の場所にあたる。

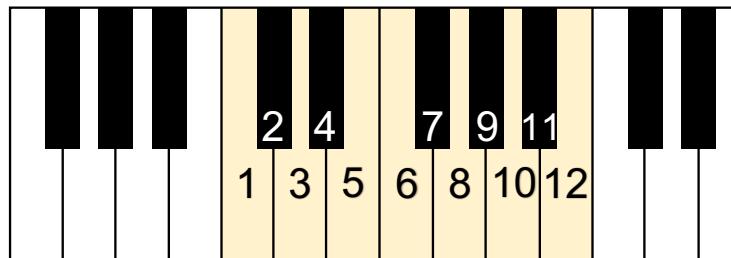


図3.2-4: 1オクターブ

その中でも音階の基準とされる音が7音あり、ピアノの鍵盤では白鍵にあたる。その7音は日本では汎用的にドレミファソラシドという言葉で表す。ドレミファソラシドはイタリア語の音名表記であり、日本語ではハニホヘトイロと表す。英語ではCDEFGABと表し、ドイツ語ではCDEFGAHと表す。ドイツ語表記は英語表記と似ているが読み方が異なり、英語表記はシー・ディー・イー・エフ・ジー・エー・ビーと発音するのに対し、ドイツ語表記はツェー・デー・エー・エフ・ゲー・アー・ハーと発音するほか、最後のシの音がBではなくHであることが特徴である。

3.2.3 高圧抽出式

音符には種類がいくつかあり、それぞれ伸ばす音の長さやタイミングが異なる。音符は符頭(たま)・符幹(ぼう)・符尾(はた)でできており、音符の種類によって形が異なる。それぞれ図3.2-5に対応し、音符によって符幹が存在しなかったり、符頭が白くなったりする。基本の音符として、全音符・4部音符・2分音符・8分音符・16分音符等がある。

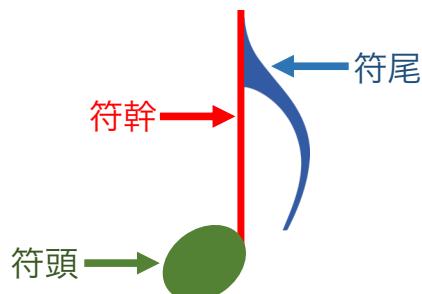


図3.2-5: 音符

全音符は音符の基準となるもので、1小節すべてを使った音を表現する。1小節とは、曲を一定の間隔で区切った範囲のことである。詳細は3.3.4の小節で説明する。そのため、音の長さは拍子によって変化する。図3.2-6のように中央が白抜きになっており、符幹がないことが特徴である。

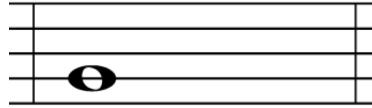


図3.2-6: 全音符

2分音符は4分音符の2つ分の長さを表現しているものである。4分の4拍子という一番基本的である拍子の単位では1小節に2つ存在することになる。図3.2-7のように、全音符に類似した符頭である。しかし、全音符にはなかった符幹があり、4分音符に類似した形をしている。

○部音符の数字の意味は全音符の何分の1の長さかで演奏するかということである。数字が大きいほど、小節ごとの音は細かくなり、1音符に対する音の長さは短くなっていく。



図3.2-7: 2部音符

4分音符はリズムをとるときなどに利用するため、覚えやすく一般的に馴染み深い長さの音符である。1小節を4つに区切った長さを表す。基本となる音符で、テンポやリズムの指標として多く用いられる。例としてメトロノーム記号という早さを表す数字は4分音符が基準となっている。

テンポ120では、4分音符1つ分0.5秒の長さにあたる。図3.2-8のように符頭と符幹があり、全音符と異なり符頭が黒いことが特徴である。

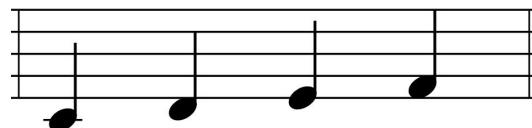


図3.2-8: 4分音符

8分音符は4分音符の半分の長さを表す音符である。4分音符1回鳴らしている間に、8分音符は2回鳴らすことができる。図3.2-9のように4分音符に符尾がついた形をしている。



図3.2-9: 8分音符

16分音符は4分音符の $1/4$ の長さを表す音符である。4分音符1回鳴らしている間に、16分音符は4回鳴らすことができる。休符を間に挟んで変則的なリズムにする場合もある。図3.2-10のように8部音符の符尾を二重にした形をしている。



図3.2-10: 16部音符

3.3 その他の記号

楽譜を構成する上で必須となる知識を説明する。

3.3.1 小節

小節は曲をある一定の間隔で区切った範囲のことである。その長さは曲によって異なり、1小節に入る音符の長さは決まっている。楽譜上では図3.3-1のように縦線で区切られる。

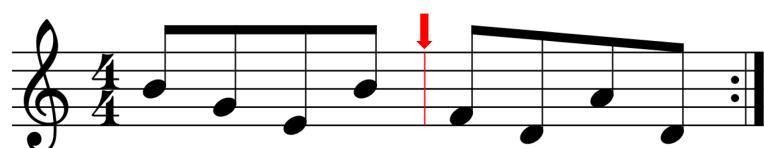


図3.3-1: 小節

3.3.2 休符

休符とは音が止んでいる長さを表す記号である。音符とは異なり、特定の位置で表示される。全休符・2分休符・4分休符・8分休符・16分休符等がある。全休符は図 3.3-2 のように表す。全ての休符の長さの基準となる休符であり、全音符と同様 1 小節休みを表現する。

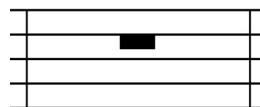


図 3.3-2: 全休符

2 分休符は 2 分音符、4 分休符は 4 分音符と同じ長さの休みを表す。その他の休符も各音符と同じ長さの休みを表している。それぞれの休符を表 3.3-1 に表す。

表 3.3-1: その他の休符

名称	休符
2 分休符	
4 分休符	
8 分休符	
16 分休符	

そして特殊な休符としては長休符が存在する。長休符は図 3.3-3 のように記述し、長休符の上の数字分の小節を休むことを意味する。

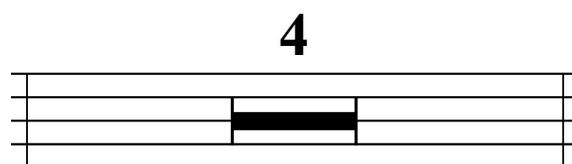


図 3.3-3: 長休符

3.3.3 拍子

拍子記号とは1小節に音符がどれくらい入るかを表している記号である。楽譜上では図3.3-4のように分数で表記されている。下の数字はどの長さの音符を基準とするか、上の数字は基準とした音符が1小節に何個入るかを表している。楽曲において最も多く用いられるのは4分の4拍子であるが、これは4分音符が1小節に4つずつ入ることを表している。小節は4小節、8小節、16小節と、4の倍数でひとまとめにされることが多い。



図3.3-4: 拍子

3.3.4 テンポ

テンポとは曲の速度を表すものである。テンポを表す際は数字を使うことが多い。この数字は1分間に同じ速さで打つ拍のことを指し、BPM (Beat Per Minuteの略)とも言われる。BPM100であれば1分間に4部音符が100拍打つということになり、BPM60であれば1分間に4部音符が60拍打つということになる。楽譜では速度表記する際、数字ではなくイタリア語を用いることもある。テンポとイタリア語表記一覧を表3.3-2で表す。

表3.3-2: テンポ表記

イタリア語表記	速さ	BPM
Largo	幅広く、ゆるやかに	40~60
Larghetto	やや遅く	60~66
Adagio	ゆっくりと	66~76
Andante	歩くような速さで	76~108
Moderato	控えめなスピードで	108~120
Allegro	快速に	120~168
Presto	急いだスピードで	168~200
Prestissimo	極めて早く	200~208

3.3.5 調号

調号とは別名調号記号といい、楽曲の調性を示す記号のことである。図 3.3-5 のように音部記号の次に変化記号というものを用いて示す。



図 3.3-5: 調号

変化記号とは、シャープやフラットと言った記号であり、それぞれ図 3.3-6 と図 3.3-7 で表す。シャープは音を半音高め、フラットは音を半音低める。楽曲には、キーというものが存在しており、それが曲の中心の音程になる。音部記号のみで何も変化記号がついていないものは、ハ長調またはイ短調と呼ばれており、シャープ記号やフラット記号が付く数によって他の長調や短調になる。



図 3.3-6: シャープ記号



図 3.3-7: フラット記号

調号は全部で 14 種類あり、シャープ系 7 種類、フラット系 7 種類で構成されている。各記号の数が異なり、その数が増加するに従って記号は右側上方か右側下方に付随していく。フラット系はシのフラットを起点に 4 度上にフラットが増えていく。シャープ系の調号はファのシャープを起点に五度上にシャープが増えていく。ここで示す 5 度、4 度とはある音からある音までの距離を表している。シャープ記号が 1 つの場合、音階のファに示されるが、2 つの場合は音階のドの位置に示される。この場合ド・シ・ラ・ソ・ファと 5 音分差があり、この差を音楽的には 5 度と表す。同じようにシャープ記号が 4 度上にあるということは、記号が増える際、4 音分離れた箇所に示すことを意味している。実際の変化を図 3.3-8 と図 3.3-9 で示す。

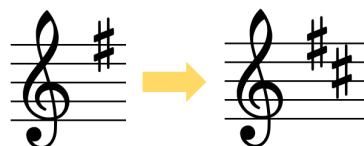


図 3.3-8: シャープ調号の変化

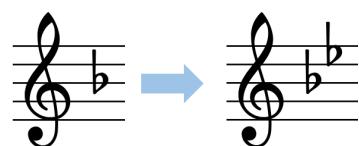


図 3.3-9: フラット調号の変化

第4章 ピアノの指導方法

本章では、ピアノの基本的な指導方法を説明する。

4.1 ピアノの基本的な指導方法

ピアノには特定の指導法はなく、指導者の方針や教室ごとにその方針は異なる。そのため、ピアノの上達は先生の指導方法によって大きく変化する。

熟練度は低くても指導が得手である指導者もいれば、熟練度が高くても指導が不得手な指導者もいる。各指導者の性格、腕前、環境など様々な要因があり、多種多様であるといえる。

4.2 個人ピアノ教室の実態

前述したようにピアノには決まった指導法がなく個人のピアノ教室となると、教室ごとの特性が顕著にみられるようになる。

しかし、指導者と生徒が1対1であることが多いため、その生徒に合わせた指導ができるることは共通している。教室によっては、生徒に合わせた曲を選択したり、学習する順番を考慮したりする場合もある。

第 5 章 画像認識ツール

本章では、楽譜を加工する画像認識とそのツールについて説明する。

5.1 画像認識とは

画像認識とは、画像や動画から特徴を抽出し、対象物を識別するパターン認識技術の 1 つである。コンピュータは人間と異なり、経験から対象が何であるか理解することが出来ない。そのため、コンピュータは大量のデータから画像に何が移っているか解析して確率的に予測する。画像認識は 1960 年頃から研究されていたが、当時のコンピュータは性能が低い上に高価であったため、大学の研究機関等しか扱えなかった。しかし、現在は電子機器が普及し、その性能も大幅に上がったため、デジタルカメラやスマートフォンなど様々な機器に画像認識機能が取り入れられている。

5.2 OpenCV

OpenCV は正式名称、Open Source Computer Vision Library と呼ばれる、オープンソースのコンピュータ・ビジョン・ライブラリのことである。2006 年 Intel によりバージョンリリースが行われ、コンピュータで画像や動画を処理することを主な目的としている。その後 Willow Garage に引き継がれ、現在は Itseez によって開発が進められている。フィルター処理や変形処理をはじめ、物体認識や機械学習等、様々な機能を利用することができる。マルチプラットフォーム対応であり、多言語による開発が可能である。OpenCV を用いて実際に顔認識と瞳認識を行った場合、図 5.2-1 のようになる。現在、プラットフォームは Windows・Linux・MacOS・Android・WindowsRT を利用でき、プログラミング言語は C・C++・Python・Java を利用できる。また、対応言語はバージョンを重ねるごとに充実しているため、今後さらに増える可能性がある。



図 5.2-1: 顔認識

第 6 章 Python について

本章では、システムを制作する上で利用するプログラミング言語について説明する。

6.1 Python とは

Python とは、プログラミング言語の 1 つである。コードがシンプルで、初心者でも扱いやすいことが特徴である。文法が単純なため、プログラムの可読性が高いことがあげられる。多くのハードウェアと OS に対応しており、オブジェクト指向・命令型・手続き型・関数型などの形式でプログラムを書くことができる。この特性から、Python は Web アプリケーション開発やデスクトップアプリケーションなどの開発をはじめ、自動処理や統計・解析など幅広く使われるようになった。プログラミング作業が容易で効率的であることから、ソフトウェア開発企業にとって時間短縮や人数削減が見込めるとして多く利用されている。近年、機械学習が多く用いられるようになり、Python が利用される場面がより多くなっている。

6.2 日本語を出力する Python ライブラリについて

本システムでは、楽譜を解析し、そのデータをもとに音階を割り出し、表示するシステムを開発している。画像開発を行っている上で利用しているのは OpenCV であるが、OpenCV は日本語表示ができないという特徴がある。そのため、プログラム上では英語表記の音階を扱わなくてはいけないが、出力する際には一般的に用いられるイタリア語表記の音階を表示しなくてはいけない。

そこで本研究では Python の画像処理ライブラリである Pillow(PIL) を用いて日本語の出力を行った。Pillow とは、PIL(Python Image Library) からフォークされた画像処理ライブラリである。OpenCV のような高度な画像処理はできないが、リサイズや回転、トリミングなどの単純な画像加工は行うことができる。実際に Pillow を用いて加工を行った画像を図 6.2-1 で示す。基本的には画像の読み込み、処理、保存に使われる他、図形の描画などを行う。日本語に対応しているため、本研究では Pillow を利用して音階の表示と画像の保存を行う。

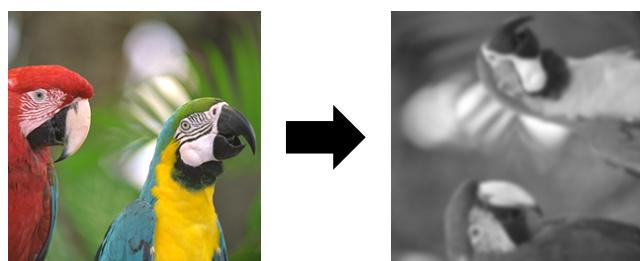


図 6.2-1: Pillow を用いて加工を行った画像

第7章 本研究で開発するシステムの概要

本章では、実際に制作するシステムについて説明する。

7.1 実装機能

本研究で実装する機能は音階の自動表示である。楽譜をスキャンし、その画像をOpenCVで解析することで楽譜の五線と音符の位置を割り出す。その座標から音階を割り出し、楽譜に表示させることを最終目的とする。本システムのフローチャートを図7.1-1に示す。

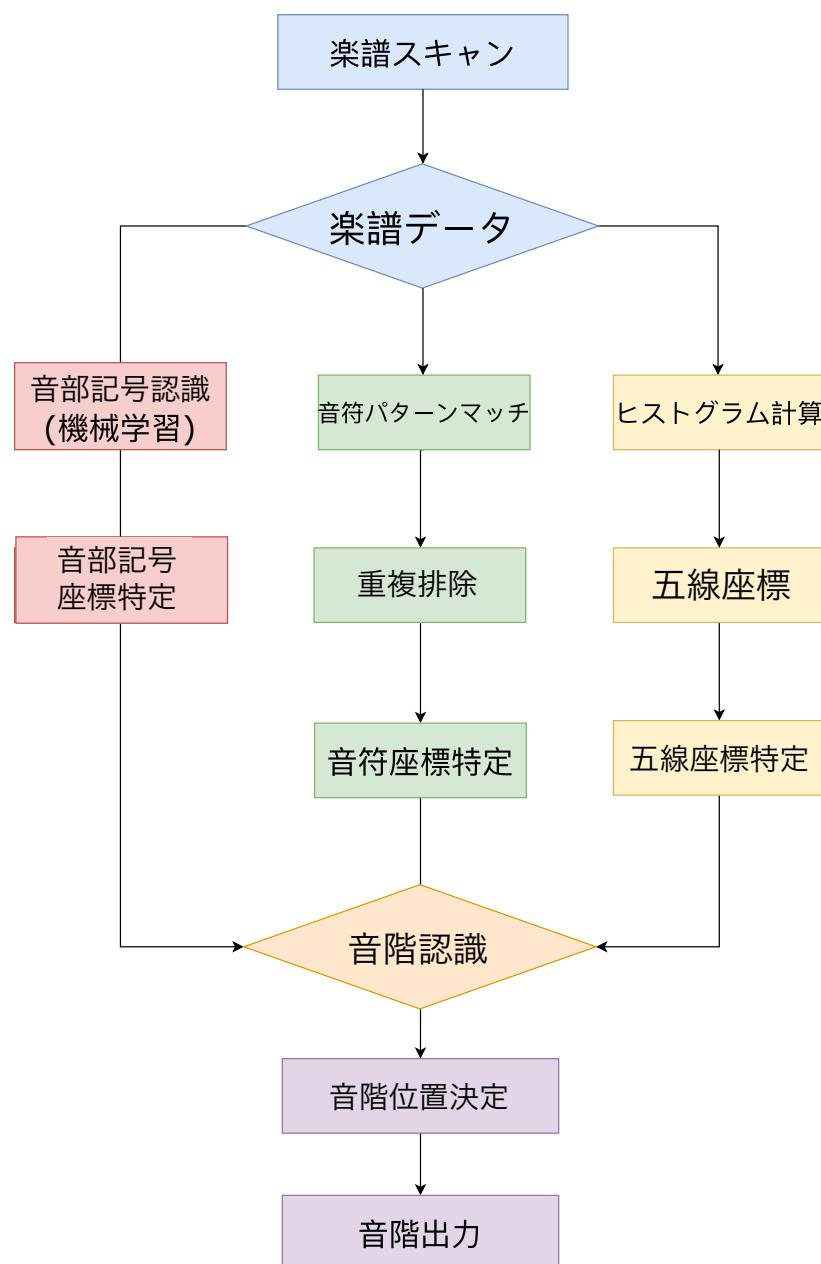


図7.1-1: 本システムのフローチャート

7.2 楽譜ならではの画像認識について

楽譜は音部記号と五線と音符で構成されている。つまり、その3要素を抽出できれば音階を割り出すことができるということになる。音部記号は機械学習を用いる。機械学習はポジティブ画像（学習対象が存在している画像）とネガティブ画像（学習対象が存在しない画像）から特徴点を抽出し、対象の画像から学習対象があるか判別するものである。TrainingAssistantというカスケード分類器を利用することでポジティブ画像、ネガティブ画像、どちらにも属さない画像（関係ない記号である等）に分類することができる。本研究では、そのデータを利用して機械学習を行う。ト音記号の特徴を学習させることで、ト音記号のみの楽譜にも、ヘ音記号が混じった楽譜にも対処することができる。

五線は各ピクセルを解析し、ヒストグラムを生成することで座標を割り出す。各X軸の黒点の数をカウントし、一定の個数以上の黒点が検出されたX軸には五線が存在している確率が極めて高いと推測される。それを利用して座標を記録する。

音符の検出にはパターンマッチを利用する。楕円の形を検出対象とし、音符の符頭を割り出す。閾値を低めに設定して検出し、多重マッチングをしている箇所を抽出することで、正確に符頭を割り出す。また、検出した楕円の中央座標を記録することで正確に音階を割り出せるようにする。

7.3 楽譜認識

OpenCVとPythonを利用して実装していく。楽譜認識は音部記号、五線、音符の順に検出していき、その情報を利用して音階の出力を試みる。

7.3.1 音部記号の認識

音部記号はト音記号とヘ音記号の2種類が主に利用される。そこで、ト音記号を対象に機械学習を行い、ト音記号のみを判別できるようにする。学習の為に用意した画像は49枚あり、その内、ポジティブ画像は25枚、ネガティブ画像は8枚、無効画像は16枚になった。ポジティブ画像から、正解ベクトルデータを作成する事ができる。以下のコマンドを実行することで、正解ベクトルデータが作成される。

```
$opencv_traincascade -data ./cascade/trained_data/ -vec ./vec/image.vec  
-bg nglis.txt -numPos ポジティブ画像の数 -numNeg ネガティブ画像の数
```

そして、作成された正解ベクトルデータと不正解画像を元で学習を行うことができる。機械学習は、以下のコマンドを用いる。

```
$opencv_createsamples -info info.dat -vec treble_clef.vec  
-num 学習に利用した画像の数 -bgcolor 255 -maxidev 40 -maxxangle 0.8 -maxyangle 0.8
```

コマンドを用いる。学習が成功すれば、図7.3-1のようにstageが実行され、xmlデータが作成される。

```

===== TRAINING 1-stage =====
<BEGIN
POS count : consumed 2400 : 2400
NEG count : acceptanceRatio 100 : 0.769231
Precalculation time: 7
+-----+
| N | HR | FA |
+-----+
| 1| 1| 1|
+-----+
| 2| 1| 1|
+-----+
| 3| 1| 1|
+-----+
| 4| 1| 1|
+-----+
| 5| 1| 1|
+-----+
| 6| 1| 1|
+-----+
| 7| 1| 0.98|
+-----+
| 8| 1| 0.98|
+-----+
| 9| 1| 0.96|
+-----+
| 10| 1| 0.96|
+-----+
| 11| 1| 0.96|
+-----+
| 12| 1| 0.96|
+-----+
END>
Training until now has taken 0 days 0 hours 3 minutes 22 seconds.

```

図 7.3-1: 機械学習

それをプログラムで読み込ませれば、図 7.3-2 のように楽譜で対象物をマークすることができる。

7.3.2 五線の検出

次は五線の検出を行う。本研究の方式では、見開きの楽譜の五線を正確に検出することができない。そのため、新しく楽譜を図 7.3-3 のように片面スキャンした。



図 7.3-2: ト音記号の認識



図 7.3-3: 片面スキャンした楽譜

図 7.3-3 の Y 座標の黒点の数をカウントし、一定の個数以上の黒点が存在する座標を五線であるとした。一定の個数以上の黒点が見られる箇所に色を付与し、表示すると図 7.3-4 のようになる。五線の位置が正確に検出されることがわかる。検出した五線の Y 座標を配列に入れることで音階の計算時に利用する。

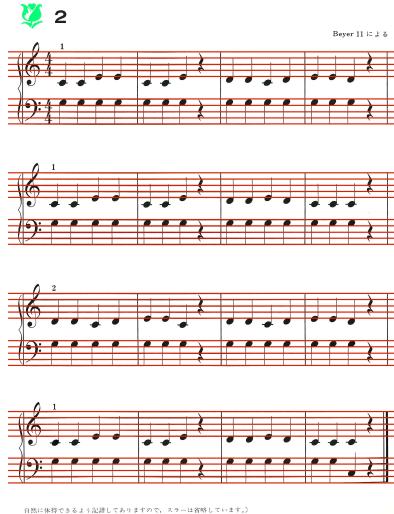


図 7.3-4: 成功した五線検出結果

7.3.3 音符の検出

最後に音符の検出を行う。本研究は音階の表示を目的としているため、音符の符頭である楕円が検出できればよい。そのため、楕円のパターンマッチを用い、楕円の検出を試みた。すると、図 7.3-5 のようにいくつも検出が重複した。

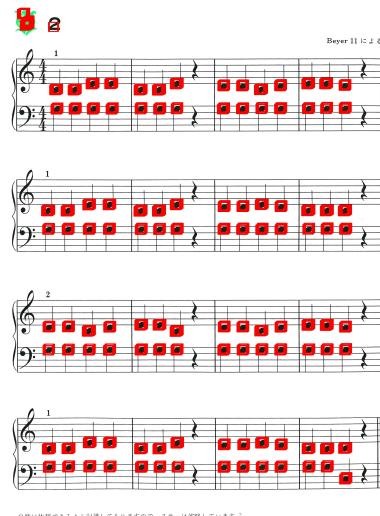


図 7.3-5: 重複した音符検出結果

図 7.3-5 のように検出が重複している場合、一定のマッチ数がある箇所は 1 つに絞ることで正確に符頭を検出できると考えた。ここで、Ruby を用い、データの精査を行う。50 個以上の重複がある検出のみを残し、表示すると図 7.3-6 のようになつた。各音符に 1 つの四角が割り当てられている。

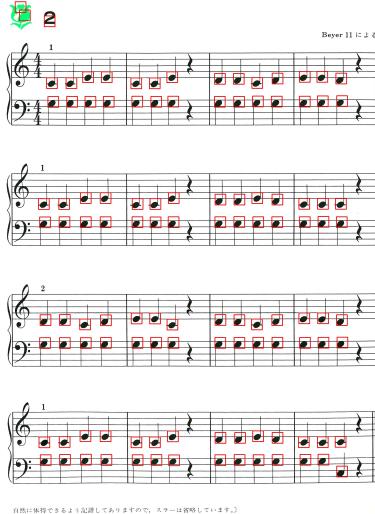


図 7.3-6: 検出を絞った出力結果

また、検出した座標は四角の左上のものであるため、中央の座標にする。四角の中央を楕円の中央であると仮定して、プログラムを変更する。そして、中央に小さな円を表示すると図 7.3-7 のようになった。すべて中央付近に表示できているため、円の座標を配列に入れる。



図 7.3-7: 音符の中央の検出結果

楽譜の上方にも検出が出ているが、音階を出力する際に無効化する。これで音階を出力するために必要な音部記号、五線、音符の座標が判明した。

7.4 音階出力

音階を求めるために必要な情報は五線の中央の座標と音符の座標、そして五線から求める 1 音階の間隔である。各音階は五線の中央（上から 3 本目の五線）を求め、そこから音符の楕円までどのくらいの距離があるか計算することで導くことができる。

7.4.1 五線の分割

まずは、前章で判明した五線の座標を読み込み、図 7.4-1 のように五線を 5 本 1 組に分割して配列に格納する。五線の中点を求めたり、各音階を計算する上で五線は必要となるため、各線を 5 本 1 組にした。

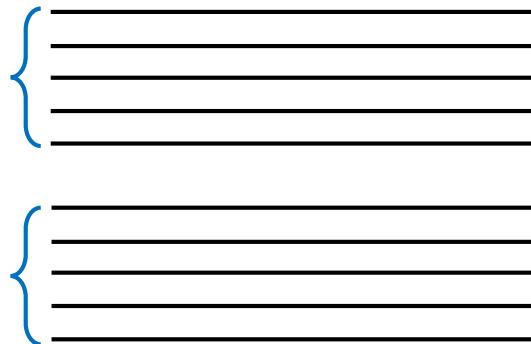


図 7.4-1: 五線を 5 本ずつに分ける

7.4.2 五線の中央の線の検出

そして、音階を求める上で重要なのは五線の中央の線である。五線の中央は各音階を求める上で基準となる。五線の中央を図 7.4-2 に示す。五線の中央は分割した五線を格納した配列の 3 番目の値を取り出すことで求めることができる。各五線の中央の線から、音符の楕円の中央までの距離を計算し、各音階の中で最も近い値の音階を出力する。

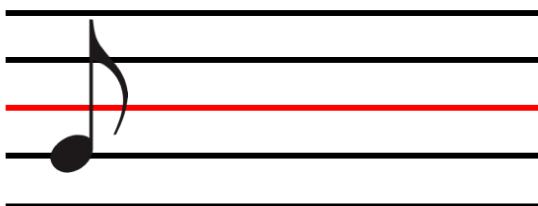


図 7.4-2: 五線の中央

7.4.3 1 音階の範囲設定

各音階を計算するにあたって必要なのは、1音階の範囲を設定することである。1音階の範囲は五線の間隔の半分に設定する。実際の1音階の範囲を図7.4-3に示す。赤い線が五線の中心の線であり、青い線が隣り合う音階の位置である。この両線の差を1音階の差であるとし、各音階を設定する基準とする。

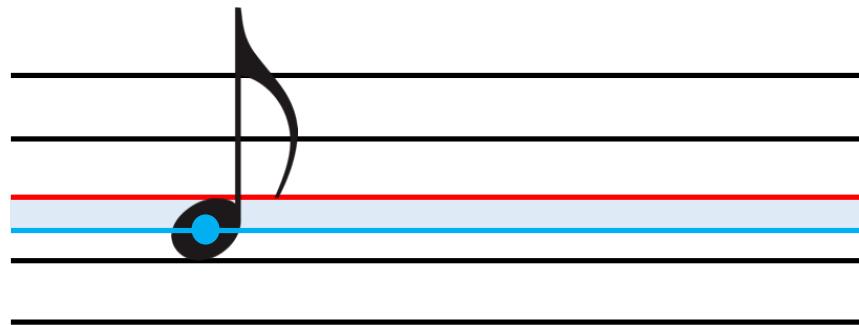


図7.4-3: 1音階の範囲

7.4.4 音階を求める

五線の中央の線と1音階の範囲が分かれれば、音階を計算することができる。Y軸から見て音符が何段階ずれているのか割り出し、その差分に応じて音階を割り当てる。五線の中央よりも上に音階がある場合は、配列に割り当てることで音階を割り出すことができるが、五線の中央よりも下に音階がある場合は数が負の数になってしまふため、配列に割り当てることができない。そのため、絶対値を用い、正しく配列に割り当てる必要がある。実際に音階に割り当てる様子を図7.4-4に示す。

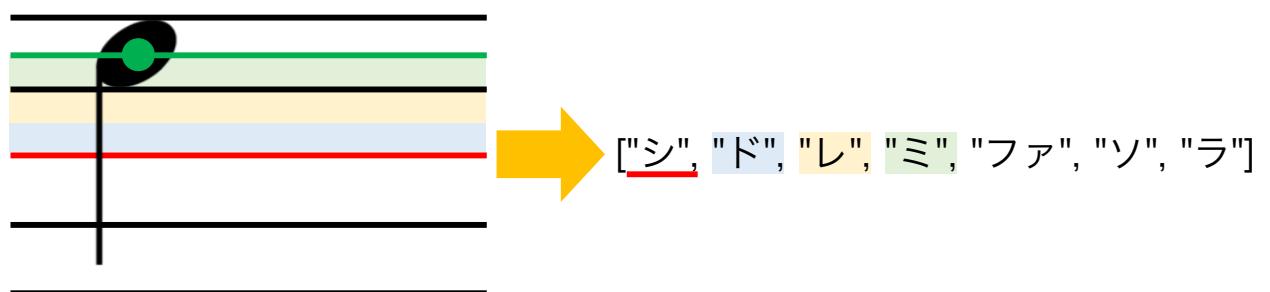


図7.4-4: 音階の求め方

7.4.5 音階の切り替え

ヘ音記号はト音記号とは別のを用意してそこに当てはめることで出力する。各音階の配列は以下のように設定する。

ト音記号→["シ", "ド", "レ", "ミ", "ファ", "ソ", "ラ"]

ヘ音記号→["レ", "ミ", "ファ", "ソ", "ラ", "シ", "ド"]

各配列は五線の中心の音階からはじめ、順に割り当てる。こうすることで各音階が正確に出力される。

7.4.6 音階を表示する

各音階を割り当てることができたので音階の出力に入る。OpenCV は日本語対応していないため、Pillow を用いて各音符の下に日本語を出力する。実際に出力した画像を図 7.4-5 に示す。



図 7.4-5: 出力画像

第8章 システムの有用性の検証

本章では、実際に制作したシステムの有用性の検証方法について説明する。

8.1 検証方法

本研究で提案・開発した音階表示システムを実際に見ていただき、本システムが時間コストの削減に有用であるか確認することを目的とする。

8.2 調査方法

1. 調査対象

ピアノの指導者、演奏者4名

2. 調査方法

加工を行っていない楽譜と、システムを用いて音階を割り振った楽譜を見ていただき、システムの妥当性や今後の展望についてアンケート調査を行う。アンケートは4段階評価(そう思う、まあそう思う、あまりそう思わない、思わない)と自由記述の計5問とする。

3. 調査期間

期間：平成30年12月20日～平成30年1月14日

場所：被験者ごとに個別対応

時間：被験者ごとに個別対応

アンケート時間：10分程度

4. 調査内容

実際に指導者に質問した内容を表8.2-1に示す。

表8.2-1: 設問内容

問	内容
1	初心者を指導する際、楽譜に音階を手書きすることで対処するか
2	経験上、音階表示が必要なのはどのくらいの期間か
3	このシステムを実際に指導する際に使用したいと思うか
4	このシステムがあることで時間的節約ができると思うか
5	初心者のためにどのような機能があれば良いと思うか

8.3 アンケート結果

ピアノ指導者、演奏者に行ったアンケート結果について示す。

現状の調査

問1、初心者を指導する際、楽譜に音階を手書きすることで対処するか



図 8.3-1: 音階を手書きで対処するかの設問結果

音階を実際に対処するかの回答を図 8.3-1 に示す。この結果から、楽譜を手書きで対処する場合もあるが、全く行わない場合もあることがわかった。個人のピアノ教室では、音階を書かない指導を行う場合もあり、その差が見られたものであると推測される。

現状の調査

問2，経験上，音階表示が必要なのはどのくらいの期間か

表 8.3-1: 問2 の回答結果

2～3年
3年
2～3ヶ月・個人差があるが半年で慣れはじめる
最初から音階を表示して演奏させない

問2の実際の回答結果を表8.3-1に示す。音階を表示するかについての回答ではばらつきが見られた。半年で慣れはじめるという意見もあれば、2～3年という意見も見られた。読譜に関しては個人差が大きく関与するため、一概に期間があるわけではないことがわかる。また、最初から音階を表示しないで指導する場合も見られた。

システムの必要性の検証

問3，このシステムを実際に指導する際に使用したいと思うか

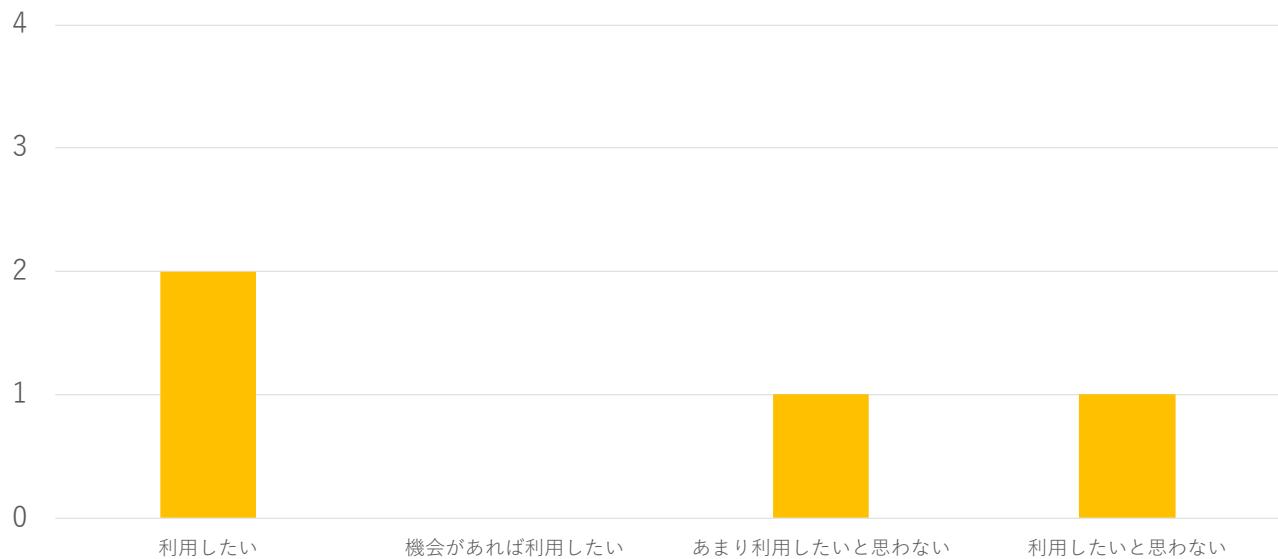


図 8.3-2: システムの必要性の設問結果

実際に指導する際に利用したいかの回答結果を図 8.3-2 に示す。利用したいという意見が半分あるものの、あまり利用したいと思わない・利用したいと思わないという意見も見られた。これは、教室ごとの指導方針によって変わると推測される。また、生徒によっては利用しない・利用するという意見も見られ、教室ごとの差に加えて生徒ごとにも導入するか、導入しないかが変化することがわかった。

システムの時間的有用性の検証

問4，このシステムがあることで時間的節約ができると思うか



図 8.3-3: 時間的有用性の設問結果

本システムが時間的節約に有用であるかの回答結果を図 8.3-3 に示す。この設問では全員から、短縮できるという意見を得ることができた。以上のことから本システムは時間的有用性があることがわかった。

今後のシステムの展望

問5，初心者のためにどのような機能があれば良いと思うか

表 8.3-2: 問5の回答結果

テンポ表記もあった方が良いと思う
譜面に書かれた音やリズムを歌ってくれる音声機能があれば良いと思う。
初心者の方には、楽器に触る事は勿論ですが、様々な音を耳でも沢山聴いていただきたい。
1オクターブごとの色分け
表示する音階の指定

問5の実際の回答結果を表8.3-2に示す。音階表示の機能の拡張や音情報の付与についての意見が多く見られた。本研究では音階の表示のみの対応なので、音階表示のパターンを変化させたり、音階表示以外の視覚サポート、音によるサポート等ができるとより良いシステムになると考えられる。

第9章 結言

音楽は趣味や仕事の面で多くの人が関わり、古代から現代まで長く続いている文化である。職種としてのみならず、趣味としても多くの人にあげられる音楽は世界共通の言語なのである。

特にピアノは歌やその他の楽器との相性が良いことから楽器としては最も多くの人に関与するものであり、その分奏者人口も多いことが挙げられる。ピアノを学ぶ上で楽譜は必要不可欠な要素であり、今後も多くの人々が楽譜と関わることになる。しかし、ピアノの楽譜は初心者にとって難しく、挫折の要因の1つとして挙げられる。

そこで本研究では、ピアノ指導者が各音符に手書きで音階を書き込む際、指導者にとって時間的コストがかかっている現状を問題点とし、その対策として実際に音階表示補助システムを開発し、ピアノ指導者に評価してもらうことを目的とした。

実際にOpenCVとPythonを利用し、音階を認識するシステムを制作した。実際に音階を表示し、ピアノ指導者の評価を得た。本システムを導入するかの差があるとはいえ、時間的コスト削減の面で本システムは大きく貢献できることがわかった。

また、今後の展望としては音階の表示のみならず、音階表示のパターンを変化、音階表示以外の視覚サポート、音によるサポート等ができるとより良いシステムになる。

謝辞

本研究の遂行及び本論文の作成にあたり，須田研究室の仲間に深く感謝の意を表します。そして，何よりも本論文の作成にあたり，多大なる御指導及び御助言を頂きました須田宇宙准教授に深く感謝の意を表します。

第 10 章 参考文献

参考文献

- [1] 総務省統計局 , “平成 23 年社会生活基本調査”, <http://www.stat.go.jp/data/shakai/2011/>
- [2] 早稲田大学理物理学部情報学科 板東慶一郎, “楽譜認識を活用した演奏支援ソフトウェア”, [file:///Users/masuda/Downloads/1g01p08220\(5\).pdf](file:///Users/masuda/Downloads/1g01p08220(5).pdf)
- [3] OpenCV team, “OpenCV”, <https://opencv.org/>
- [4] Python Software Foundation, “Python”, <https://www.python.org/>
- [5] 神奈川工科大学 情報学部 情報工学科 信号処理応用研究室 , “標準画像／サンプルデータ”, http://www.ess.ic.kanagawa-it.ac.jp/app_images_j.html#image_dl
- [6] paulrosen , “abcjs デモページ”, <https://abcjs.net/abcjs-editor.html>
- [7] KoheiShitaune, “TrainingAssistant”, <https://github.com/shkh/TrainingAssistant>
- [8] 教育芸術社 市川都志春 編著, “子どものバイエル第 1 集”, 2004 年 10 月 30 日 初版発行, 2018 年 3 月第 34 版発行, pp5
- [9] 株式会社ドレミ楽譜出版社 橋本晃一 編著, “中級レベルで弾けるクラシック名曲ピアノ曲集”, 1994 年 1 月初版発行, 2007 年 2 月 20 日第 7 版発行, pp26-27

付録A 作成したプログラム

Square.py

```
1 # coding:utf-8
2 import cv2
3 import math
4 import os
5 import gosen_bunkatu as gb
6 from decimal import Decimal, ROUND_HALF_UP
7 import onnkai
8 import cv2 as cv
9 import sys
10 import numpy as np
11 import PIL.Image
12 import PIL.ImageDraw
13 import PIL.ImageFont
14 from PIL import Image
15 import matplotlib.pyplot as plt
16
17 Dp = []
18
19 #音符を四角で囲う
20 def draw_square(Up, img):
21     template_width = 65
22     template_height = 65
23     for l in Up :
24         cv2.rectangle(img, (l[0], l[1]), (l[0] + template_width,
25                               l[1] + template_height), (0, 0, 255), 3)
26         Dp.append([int(l[0] + (template_width)),
27                    int(l[1] + (template_height))])
28
29 #画像の読み込み・加工
30 img = cv2.imread('image1025.png')
```

```
31 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
32 retval, binarized = cv2.threshold(gray, 224, 255,
33                                     cv2.THRESH_BINARY_INV)
34
35 #玉の検出座標データを読み込む
36 f = open("output2.dat", "r")
37
38 #座標を入れるための配列を作る
39 Up = []
40
41 #データをすべて書き出して int 型にする
42 for x in f:
43     temp = x.replace('\n', '')
44     temp1 = temp.replace('\r', '')
45     temp2 = temp1.split(" ")
46     Up.append( [ int(temp2[0]), int(temp2[1]) ] )
47
48 draw_square(Up, img)
49 cv2.imshow('score', img)
50 cv2.imwrite('sikakuhyouji.png', img)
51 cv2.waitKey(0)
52 cv2.destroyAllWindows()
```

Circle.py

```
1 # coding:utf-8
2 import cv2
3 import math
4 import os
5 import gosen_bunkatu as gb
6 from decimal import Decimal, ROUND_HALF_UP
7 import onnkai
8 import cv2 as cv
9 import sys
10 import numpy as np
11 import PIL.Image
12 import PIL.ImageDraw
13 import PIL.ImageFont
14 from PIL import Image
15 import matplotlib.pyplot as plt
16
17 Dp = []
18
19 #音符の中央に円を描画
20 def draw_circle(Up, img):
21     template_width = 50
22     template_height = 80
23     for l in Up:
24         cv2.circle(img, (l[0] + (template_width / 2),
25                         l[1] + (template_height / 2)), 12,
26                     (255, 255, 0), thickness = -1)
27     Dp.append([int(l[0] + (template_width / 2)),
28                int(l[1] + (template_height / 2))])
29
30 #画像の読み込み・加工
31 img = cv2.imread('image1025.png')
32 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
33     retval, binarized = cv2.threshold(gray, 224, 255, cv2.THRESH_BINARY_INV)
34
35     #玉の検出座標データを読み込む
36     f = open("output2.dat","r")
37
38     #座標を入れるための配列を作る
39     Up = []
40
41     #データをすべて書き出して int 型にする
42     for x in f:
43         temp = x.replace('\n', '')
44         temp1 = temp.replace('\r', '')
45         temp2 = temp1.split(" ")
46         Up.append( [ int(temp2[0]), int(temp2[1]) ] )
47
48     draw_circle(Up, img)
49     cv2.imshow('score', img)
50     cv2.imwrite('maruhyouji.png', img)
51     cv2.waitKey(0)
52     cv2.destroyAllWindows()
```

Decision.rb

```
1  #!/usr/local/bin/ruby
2
3  #重複した音符の検出データを1つにまとめる
4  def near?( x, y, box )
5      box.each do | b |
6          if(( ( b[0] - x ).abs < 50 ) && ( ( b[1] - y ).abs < 50 ) )
7              return true
8          end
9      end
10     return false
11 end
12
13 box = [];
14
15 ARGF.each_line do | line |
16     x, y = line.split
17     nx = x.to_i
18     ny = y.to_i
19
20     if( near?( nx, ny, box ) )
21     else
22         box.push( [ nx, ny ] )
23         puts x + " " + y
24     end
25 end
```

Line.py

```
1 # coding:utf-8
2 import cv2
3 import math
4 import numpy
5 import os
6
7 image = cv2.imread('image1025.png')
8 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
9 retval, binarized =
10         cv2.threshold(gray, 224, 255, cv2.THRESH_BINARY_INV)
11
12 # 空の配列を作る
13 hist = []
14
15 # 3本ずつ見ていく
16 for y in range( 1, binarized.shape[0] - 1 ):
17     h = 0
18     pu = binarized[y-1]
19     pc = binarized[y]
20     pd = binarized[y+1]
21     for x in range( 1, binarized.shape[1]-1 ):
22         dx = max( abs( int(pc[x]) - int(pc[x-1])), 
23                     abs( int(pc[x]) - int(pc[x+1])) )
24         dy = max( abs( int(pc[x]) - int(pu[x]) ), 
25                     int(pc[x]) - int(pd[x]) )
26         if( dy>32 and dy > dx*4 ):
27             h=h+1
28     # 黒の点の数をカウント
29     hist = hist + [h]
30
31 y_line =[]
32
```

```
33 # 黒の点の数に合わせて線を塗る
34 for y in range( 1, binarized.shape[0]-3 ):
35     #print( "y=", y, ":" , hist[y] )
36     if( hist[y] > 500 ):
37         if (len(y_line) is 0):
38             y_line.append(y)
39             cv2.line(image, (0,y),(binarized.shape[1],y),
40                     (0,32,224), 5)
41         elif (y_line[len(y_line) - 1] + 10 < y):
42             y_line.append(y)
43             cv2.line(image, (0,y), (binarized.shape[1],y),
44                     (0,32,224), 5)
45
46 print ( y_line )
47 cv2.imshow('image', image )
48 cv2.imwrite('line2.png', image)
49 cv2.waitKey(0)
50 cv2.destroyAllWindows()
```

index.py

```
1 # coding:utf-8
2 import cv2
3 import math
4 import os
5 #gosen_bunkatu.py を呼び出している
6 import gosen_bunkatu as gb
7 #四捨五入するためのライブラリ
8 from decimal import Decimal, ROUND_HALF_UP
9 #onnkai.py を呼び出している
10 import onnkai
11 import cv2 as cv
12 import sys
13 #以下日本語を表示するために必要なライブラリ
14 import numpy as np
15 import PIL.Image
16 import PIL.ImageDraw
17 import PIL.ImageFont
18 from PIL import Image
19 import matplotlib.pyplot as plt
20
21 #楕円の中央の座標を入れるための配列を作る
22 Dp = []
23
24 #音符の中央に円を描画
25 def draw_circle(Up, img):
26     template_width = 50
27     template_height = 80
28     for l in Up:
29         Dp.append([int(l[0] + (template_width / 2)),
30                    int(l[1] + (template_height / 2))])
31
32 #画像の読み込み・加工
```

```

33 img = cv2.imread('image1025.png')
34 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
35 retval, binarized = cv2.threshold(gray, 224, 255,
36                                     cv2.THRESH_BINARY_INV)
37
38 #玉の検出座標データを読み込む
39 f = open("output2.dat","r")
40
41 #楕円の座標を入れるための配列を作る
42 Up = []
43
44 #音符の座標のデータをすべて書き出して int 型にする
45 for x in f:
46     temp = x.replace('\n', '')
47     temp1 = temp.replace('\r', '')
48     temp2 = temp1.split(" ")
49     Up.append( [ int(temp2[0]), int(temp2[1]) ] )
50
51
52 #空の配列を作る
53 hist = []
54
55 #3本ずつ見ていく
56 for y in range( 1, binarized.shape[0] - 1 ):
57     h = 0
58     pu = binarized[y-1]
59     pc = binarized[y]
60     pd = binarized[y+1]
61     for x in range( 1, binarized.shape[1]-1 ):
62         dx = max( abs( int(pc[x]) - int(pc[x-1])), 
63                    abs( int(pc[x]) - int(pc[x+1])) )
64         dy = max( abs( int(pc[x]) - int(pu[x]) ), 
65                    int(pc[x]) - int(pd[x]) )
66         if( dy>32 and dy > dx*4 ):
67             h=h+1

```

```

68     #黒点の数をカウント
69     hist = hist + [h]
70
71 y_line = []
72
73 #黒点の数に合わせて線を塗る
74 for y in range( 1, binarized.shape[0]-3 ):
75     if( hist[y] > 500 ):
76         if (len(y_line) is 0):
77             y_line.append(y)
78         elif (y_line[len(y_line) - 1] + 10 < y):
79             y_line.append(y)
80
81
82 def scaleFunction(gosen, onnpu):
83     # ある音階から次の音階までの距離の計算
84     DIFF = (gosen[1] - gosen[0]) / 2 - 0.1
85
86     #gosen_bunkatu.py の関数を呼び出して 5 線を 5 本ずつに分割する
87     gosen_d = gb.splitStaff(gosen)
88
89     #五線の中央の線 (上から 3 番めの線) を配列に格納する
90     gosen_dd = []
91     for g in gosen_d:
92         gosen_dd.append(g[2])
93
94     #onnkai.py の関数を呼び出して五線譜に対する五線上の音符を分ける
95     onnkai_d = onnkai.makeScore(gosen_dd, onnpu)
96
97     #割り振った音階を格納する配列
98     assignment = []
99
100    ret_data = []
101    for i, od in enumerate(onnkai_d):
102        #五線の y 座標の配列を代入する

```

```

103     gosen_y = od["gosen"]
104     #へ音記号とト音記号の切り替えを行う
105     cases = None
106     if i % 2 is 0:
107         __format = ["シ", "ド", "レ", "ミ", "フ      ア", " "
108             ソ", "ラ"] #表示する音階のデータ
109     else:
110         __format = ["レ", "ミ", "フ      ア", "ソ", "ラ", " "
111             シ", "ド"] #表示する音階のデータ
112
113     for note in od["note"]:
114         #音符のy軸取得
115         note_y = note[1]
116         #音符を取るときの誤差の調整
117         normalize = -1
118         #y軸から見て音符が何段階ずれているか
119         n = (gosen_y - note_y - normalize) / DIFF
120         #以下2行で上の値を四捨五入
121         dec = Decimal(str(n))
122         n = int(dec.quantize(Decimal('0'), rounding=ROUND_HALF_UP))
123         #以下でassignmentにpythonオブジェクトを格納する
124         data = {"score_line": gosen_y, "n": note}
125         #四捨五入した値が自然数なら__format[scale]番目を
126         #data["scale"]に代入
127         if n >= 0:
128             scale = n % len(__format)
129             #自然数でなければ__formatの(7 - ((絶対値n mod 7)+ 1) mod 7
130             番目の値をdata["scale"]に代入
131         else:
132             scale = (abs(n) % len(__format)) + 1
133             scale = (len(__format) - scale + 1) % len(__format)
134
135         data["scale"] = __format[scale]
136         #dataの値をそれぞれの変数に格納する
137         assignment.append(data)

```

```

134         ret_data.append(data)
135     return ret_data
136
137 #OpenCV が日本語対応していないため、Pillow というライブラリを利用する
138 def draw_text_by_jp(CV2PIL_normalize, coordinate, scale):
139     #色置換をした画像を変数 draw に代入する
140     draw = PIL.ImageDraw.Draw(CV2PIL_normalize)
141     #フォント指定
142     font_ttf = '/System/Library/Fonts/ヒラギノ角ゴシック W3.ttc'
143     #フォントサイズ指定、変数 font に代入する
144     font = PIL.ImageFont.truetype(font_ttf, 40)
145     #テキスト表示
146     draw.text((coordinate[0], coordinate[1]), scale.decode('utf_8'),
147                 font=font, fill=(0, 0, 0))
148     return CV2PIL_normalize
149
150
151 if __name__=='__main__':
152     #五線の y 座標を gosen に代入している
153     gosen = (y_line)
154     #音符の楕円の関数を呼び出している
155     draw_circle(Up, img)
156     #音符の座標を onnpu に代入している
157     onnpu = (Dp)
158     #scaleFunction という関数を呼び出している
159     ret = scaleFunction(gosen, onnpu)
160     #OpenCV の色の置換を行っている
161     CV_im_RGB = img[:, :, ::-1].copy()
162     CV2PIL_normalize=Image.fromarray(CV_im_RGB)
163     #英語を日本語に置換している
164     convert = {"A": "ラ", "B": "シ", "C": "ド", "D": "レ",
165                "E": "ミ", "F": "ファ", "G": "ソ"}
166
167     #配列がある限り繰り返す

```

```
168     for i, r in enumerate(ret):
169         output = []
170         for key, value in r.items():
171             output.append(value)
172
173     #音階データがある限り音階表示をする
174     for r in ret:
175         x = r["n"][0]
176         y = r["score_line"]
177         scale = r["scale"]
178         #楽譜の指定の位置に音階を表示している
179         coordinate = (x - 20, y + 140)
180         draw_text_by_jp(CV2PIL_normalize, coordinate, scale)
181
182     #画像を表示する
183     plt.imshow(CV2PIL_normalize)
184     plt.show()
185     #画像を保存している
186     CV2PIL_normalize.save('gazou5.png')
```