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

図目次

表目次

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ったが，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．

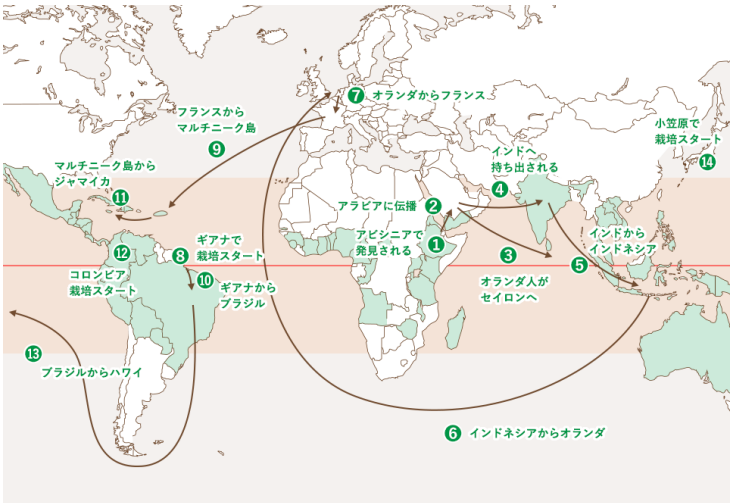

図 2.2-1: UCC ホームページより:コーヒーの軌跡

2.3　コーヒーの流行について

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．

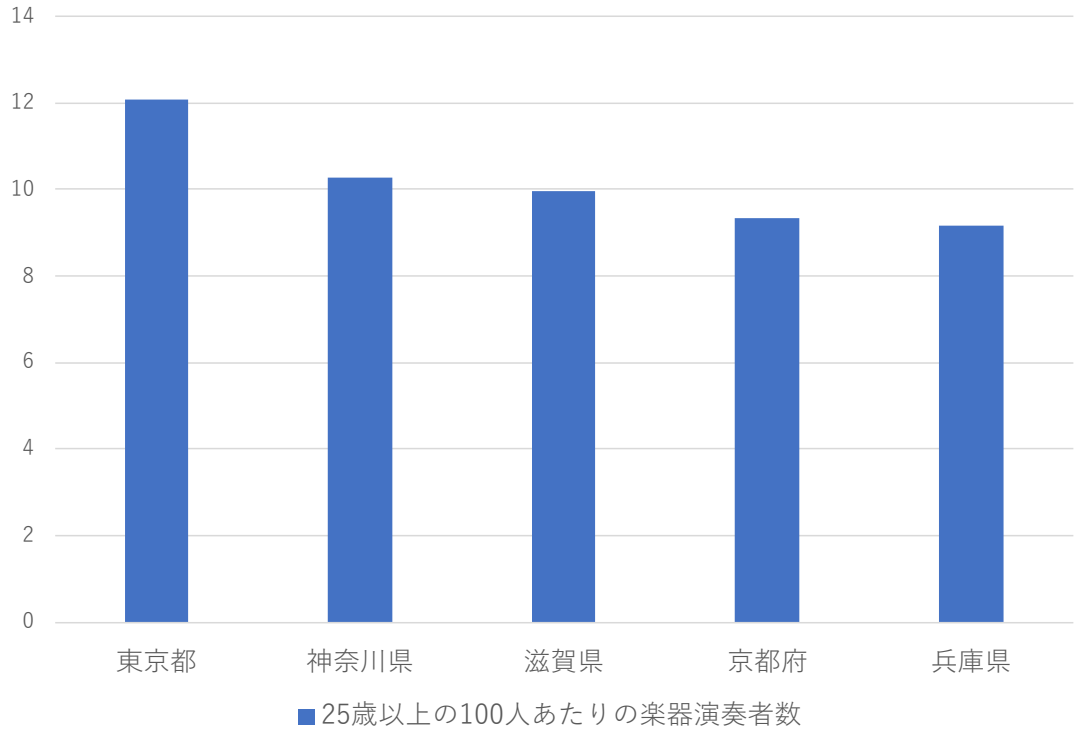

図 2.3-1: 楽器演奏人口の上位 5 都府県

2.3.1　ファーストウェーブ

2.3.2　セカンドウェーブ

2.3.3　サードウェーブ

第 3 章　抽出器具について

本章では，基本的な抽出器具等を説明する．

3.1　抽出器具

抽出器具とは，粉末状となったコーヒー豆から，味や成分を抽出する道具のことを指す．それぞれが多種多様な特徴を持っており，同じコーヒー豆を使っていても，使用した器具によって風味は大きく変化する．

コーヒーは，使用する抽出器具・方式によって大きく味を変える．コーヒープレスに代表される浸漬式は，味が濃厚に抽出される反面，出来上がりの粉末感が強くなる．プアオーバーに代表される透過式は，ペーパーフィルターを使用する場合，鮮やかでさっぱりとした味わいを表現できるが，抽出する人によって味が大きく変わることが難点である．エスプレッソに代表される高圧抽出では，とても濃厚な味が抽出されるが，タンピングなどの独自のテクニックを習得する必要がある．上記に挙げた 3 つの他にも抽出方法は多数存在し，手順やテクニックも異なるため，利用者の練度を問わず，個別の学習が必須の分野になっている．

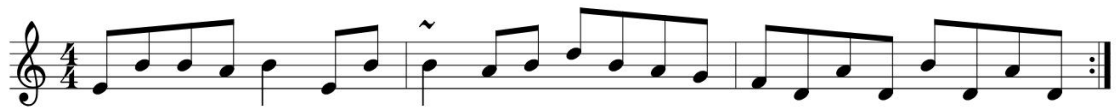

図 3.1-1: 五線譜の一例

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．

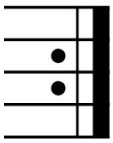

図 3.1-2: 終止符

3.1.1 プアオーバー

プアオーバーは，別名ハンドドリップとも呼ばれており，日本の喫茶店で多く提供されてきたことから，日本人に特に馴染みの深い抽出器具と言える．フィルターの上に挽いた豆を置き，お湯を回しかける事によって，下部のグラスサーバーにコーヒーを抽出していく．フィルターを介しているため，出来上がりは，粉末感は殆どなくクリーンな味わいになりやすい．また，お湯の注ぐタイミングや蒸らしの時間の掛け方などにより，様々な流派が存在し，淹れた人によって大きく風味が変化する事も特徴の一つである．

3.1.2 コーヒープレス

コーヒープレスは，フレンチプレスと呼ばれる器具に，挽いたコーヒー豆とお湯を入れ，時間をかけて抽出を行う器具である．紅茶でいうティープレスのように，抽出時間経過後は上からステンレスフィルターを押し下げ，豆と液体を分離することによって，抽出を完了させる．仕上がりは．多少の粉末感を残すものの，濃厚でまろやかな味わいである．ステンレスフィルターを使用するため，コーヒーのオイルが吸収されず，コーヒー豆本来の味が楽しめることも利点の一つである．

3.1.3 エスプレッソ

エスプレッソは，イタリアを発祥とする飲み方であり，水蒸気やピストンなどで圧力をかけ，短い時間で抽出されたコーヒーのことを指す．自動式・半自動式・ピストン式など様々なエスプレッソマシンが存在するが，使用者自身でフィルターに豆を押し込み，機械によって気圧をかける半自動式が最も一般的である．仕上がりは極めて濃厚であり，そのままの状態で飲む以外にも，ミルクを追加しカフェラテ・カプチーノとして提供されることが多い．

3.1.4 ソロフィルター

ソロフィルターは，カフェ等ではあまり提供されないが，コーヒーを初めて淹れる人でも簡単に抽出でき，入門用に最適な器具の一つである．ステンレスフィルターの上に挽いたコーヒー豆を置いた後，複数の極細の穴があるパーツに規定量のお湯を注ぐだけで，適切な湯量が豆に注がれ続ける．容量は 1 杯分のコーヒー豆しか入らないため，大人数分のコーヒーを抽出するには不向きだが，粉末感の殆どない高品質なコーヒーを手軽に味わえるため，コーヒーを学び始めの人や，朝の忙しい時間でも簡単に抽出することができる．

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．

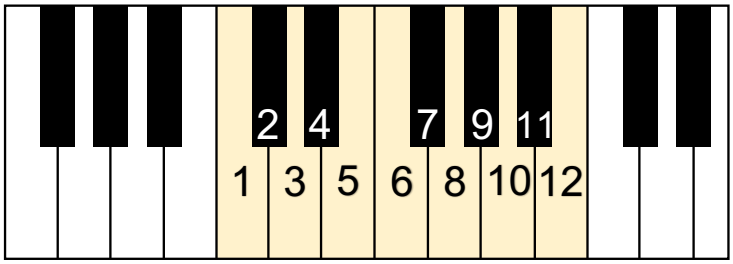

図 3.2-4: 1 オクターブ

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．

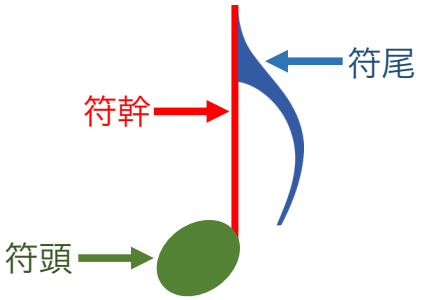

図 3.2-5: 音符

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．

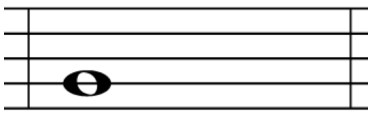

図 3.2-6: 全音符

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．

〇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．

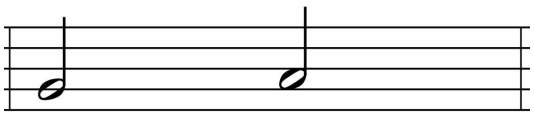

図 3.2-7: 2 部音符

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．

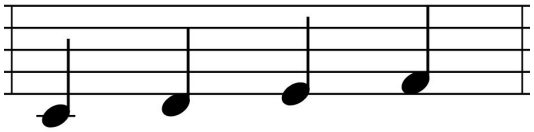

図 3.2-8: 4 分音符

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．

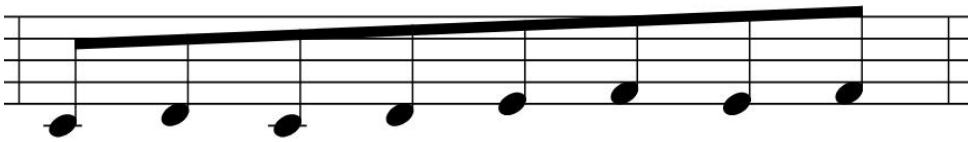

図 3.2-9: 8 分音符

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．

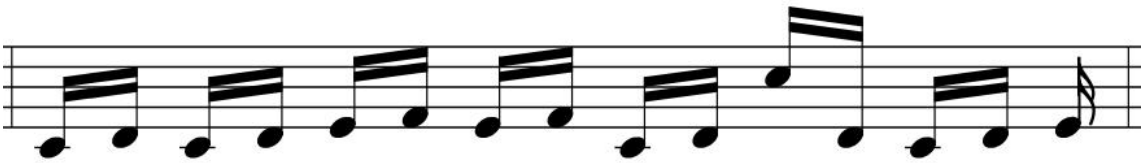

図 3.2-10: 16 部音符

第 4 章　コーヒー抽出の学習方法

本章では，ピアノの基本的な指導方法を説明する．

4.1　抽出の基本的な学習方法

ピアノには特定の指導法はなく，指導者の方針や教 k 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　コーヒーの学習が可能な **Web** サイトついて

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．

4.3　既存の音声認識可能な **Web** レシピについて

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　音声認識ツール

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　音声認識とは

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　WebSpeechAPI

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　プログラミング言語について

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　HTML・CSS 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　音声認識を可能にする JavaScript 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．

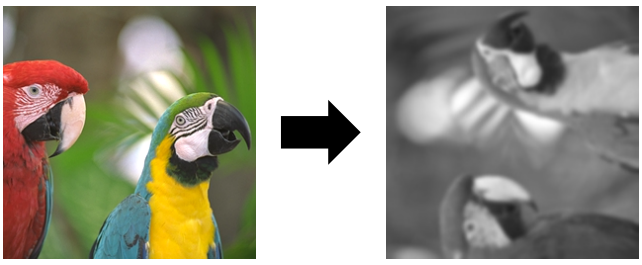

図 6.2-1: Pillow を用いて加工を行った画像

第 7 章　本研究で開発する Web レシピの概要

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．

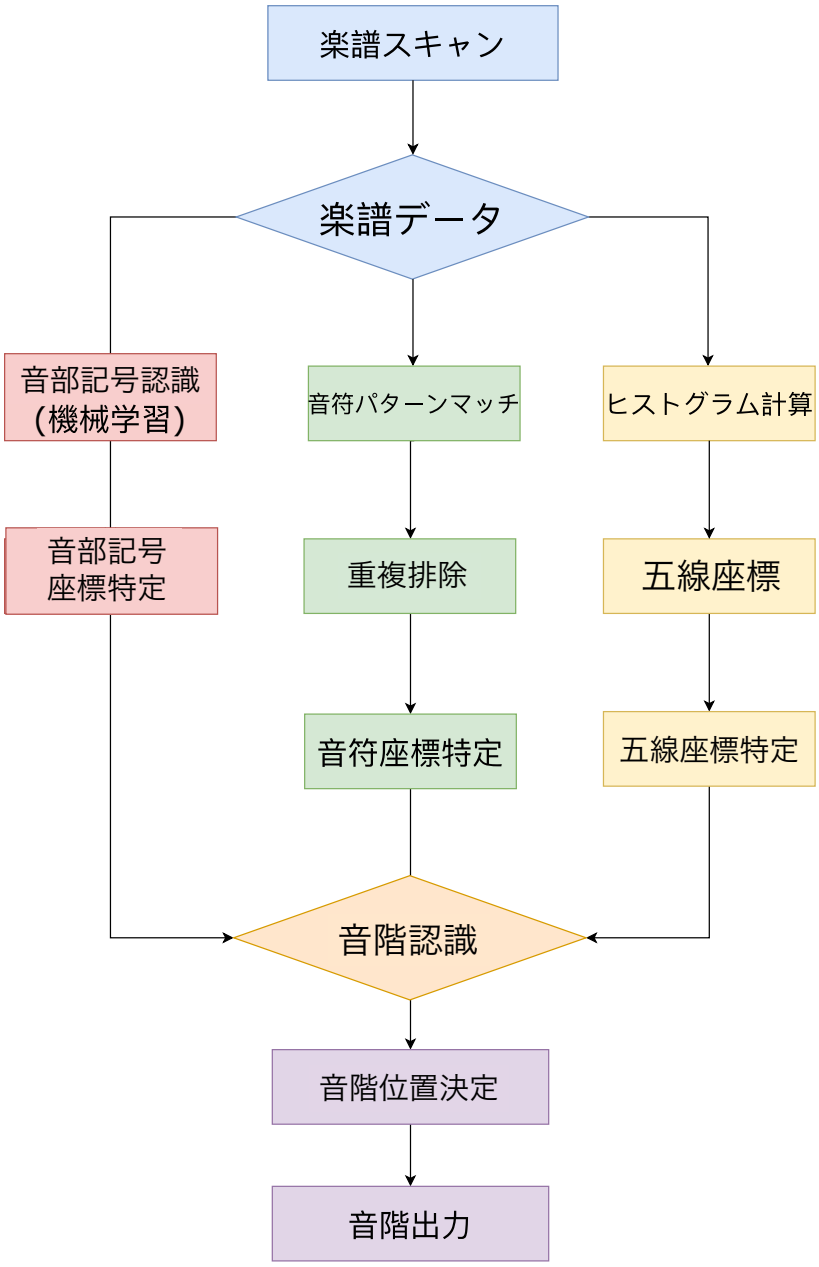

図 7.1-1: 本システムのフローチャート

7.2 本 Web レシピのページ構成について

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 (学習対象が存在してない画像) 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，そのデータを利用することで機械学習を行う．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．

7.2.1 抽出器具選択画面

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．

7.2.2 レシピ閲覧画面

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　音声認識

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．

7.3.1　音声認識の仕組み

まずは，前章で判明した五線の座標を読み込み，図**??**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．

7.3.2　音声認識によるページ移動

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 7.3-1 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．

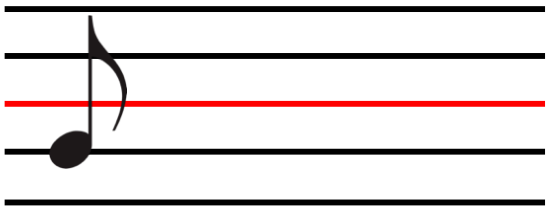

図 7.3-1: 五線の中央

第 8 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 9 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] 早稲田大学理工学部情報学科　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

[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，“こどものバイエル第 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             #ヘ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　　ァ", "
  ソ", "ラ"] #表示する音階のデータ
108             else:
109                 __format = ["レ", "ミ", "フ　　ァ", "ソ", "ラ", "
  シ", "ド"] #表示する音階のデータ
110
111             for note in od["note"]:
112                 #音符のｙ軸取得
113                 note_y = note[1]
114                 #音符を取るときの誤差の調整
115                 normalize = -1
116                 #ｙ軸から見て音符が何段階ずれているか
117                 n = (gosen_y - note_y - normalize) / DIFF
118                 #以下２行で上の値を四捨五入
119                 dec = Decimal(str(n))
120                 n = int(dec.quantize(Decimal('0'), rounding=ROUND_HALF_UP))
121                 #以下で assignment に python オブジェクトを格納する
122                 data = {"score_line": gosen_y,"n": note}
123                 #四捨五入した値が自然数なら__format[scale] 番目を
  data["scale"] に代入
124                 if n >= 0:
125                     scale = n % len(__format)
126                 #自然数でなければ__format の (7 - ((絶対値 n mod 7)+ 1) mod 7
  番目の値を data["scale"] に代入
127                 else:
128                     scale = (abs(n) % len(__format)) + 1
129                     scale = (len(__format) - scale + 1) % len(__format)
130
131                 data["scale"] = __format[scale]
132                 #data の値をそれぞれの変数に格納する
133                 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             #scaleFunctiom 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             covert = {"A": "ラ","B": "シ","C": "ド","D":
165                     "レ","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')
```