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 %前後を占めている．アラビカ種は，スペシャルティコーヒーと呼ばれる高品質コーヒーの主要製品であり，多くのカフェで提供されている．残りの 40 %はロブスタ種が占めており，こちらは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やインスタントコーヒー等に用いられる．

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．

2.3　コーヒー (ロブスタ種) の起源と伝搬の歴史

図 2.3-1: UCC ホームページより:コーヒーの軌跡

2.4 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 2.4-1 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．

2.4.1 ファーストウェーブ

2.4.2 セカンドウェーブ

2.4.3 サードウェーブ

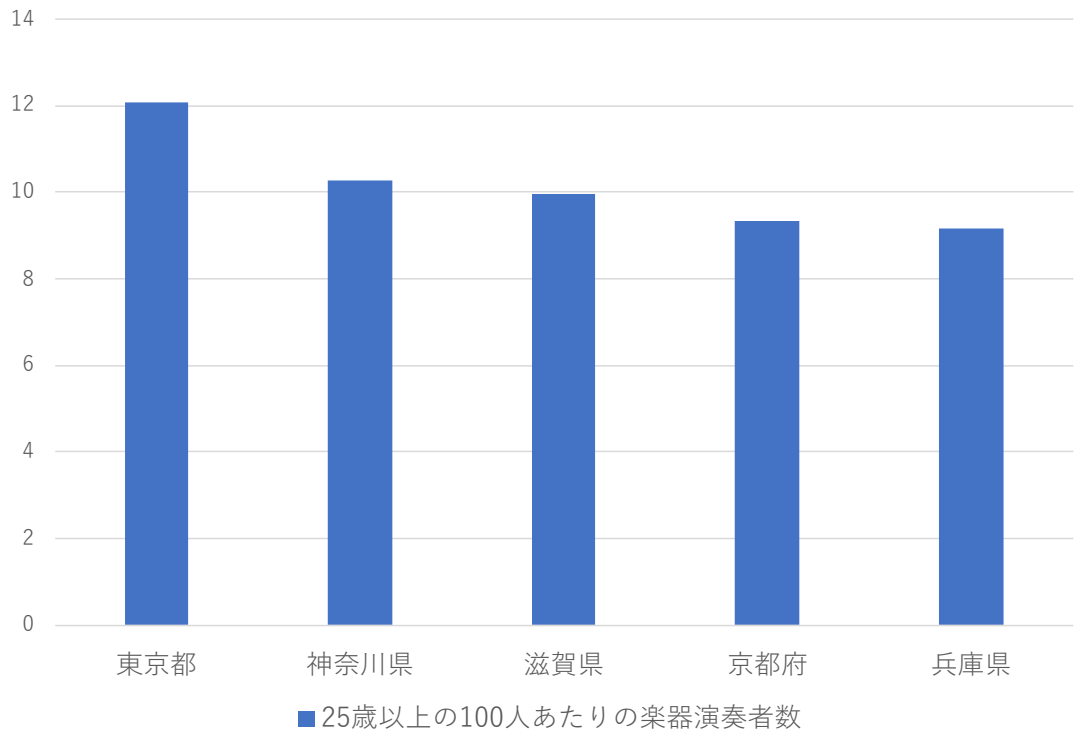

図 2.4-1: 楽器演奏人口の上位 5 都府県

第 3 章　抽出方法と抽出器具について

本章では，基本的な抽出方法や抽出器具を説明する．

3.1　抽出方式

ここでは，抽出の仕組みとも言える抽出方式を説明する．

3.1.1　浸漬式

浸漬式は，コーヒープレスやコールドブリューに代表される抽出方式である．名前の示す通り，コーヒー豆をお湯に浸し，適切な時間漬けておくことで，風味を引き出す．コーヒーは抽出を過剰に行った場合，雑味やエグ味まで抽出されてしまうため，抽出にかかる時間を適切に管理することが，浸漬式では特に重要である．比較的長時間の抽出になるため，粗挽きの豆を使用するのが一般的である．

3.1.2　透過式

透過式とは，プアオーバーやケメックスに代表される抽出方式である．浸漬式とは異なり，抽出されたコーヒーはフィルターを介し，豆とお湯が分離されサーバー

に落とされていく.

3.1.3 高圧抽出式

3.1.4 真空濾過式

3.2 抽出器具

ここでは，数多く存在すつ抽出器具の一例を説明する．

3.2.1 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．

3.2.2 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．

3.2.3 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．

3.2.4 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．

3.2.5 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．

3.2.6 サイフォン

サイフォンは，科学の実験器具のような形状をした抽出器具であり，数ある抽出器具の中でも特に見栄えの良い器具として，注目を集めている．お湯を熱したことによる蒸気圧を使用し，真空に近い状態を作り出すことで，コーヒーが器具の内部を上下し，フィルターを通して濾過される．日本の喫茶店でも多く取り入れられていたため，馴染みの深い抽出器具でもある．従来は加熱にアルコールランプを使用していたが，最近では電気式が主流となっている．

3.2.7 クレバードリッパー

クレバードリッパーは，近年新たに提案された浸漬式と透過式を合わせた抽出器具である．コーヒープレスと同様に，コーヒー豆をお湯に浸し適切な時間をおいた後，ペーパーフィルターを通し，豆とお湯を分離する抽出方法をとる．味わいは，浸漬式のマイルドさと，透過式のスッキリさを併せ持っている．一度に多くのコーヒーを抽出できるほか，プアオーバーのような複雑な工程を踏まないため，比較的簡単に抽出を行うことができる．

3.2.8 エアロプレス

エアロプレスは，その名の通り空気圧を使用した抽出器具である．開発されたのは 2005 年であり，数多くの抽出器具の中でも特に直近に提案された器具である．注射器のような形状をしており，利用者自身が器具を押し込むことによって，空気圧を調整し抽出を行う．抽出時間が短いこと，味わいが濃厚であることが特徴である．2005 年という直近の開発でありながら，様々なカフェで提供されている他，エアロプレスに特化した，抽出技術を競う大会が開催されたりと，シェアを順調に伸ばしている．

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　プログラミング言語と音声認識ついて

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．

5.1　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．

5.2　音声認識とは

5.2.1　音声認識の仕組み

5.2.2　WebSpeechAPI

5.2.3　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**??**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．

5.2.4　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**??**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．

第 6 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．

6.1　コンセプトについて

6.2　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 6.2-1 に示す．

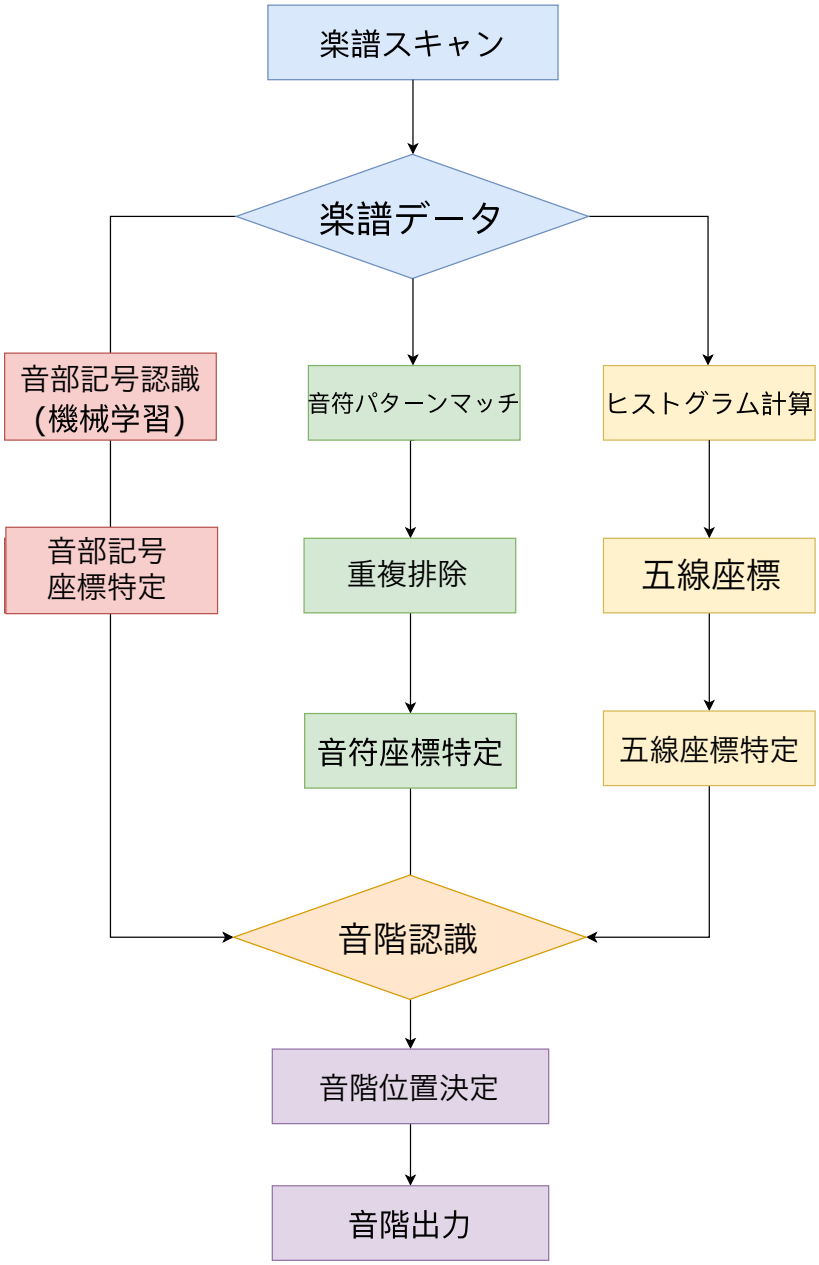

図 6.2-1: 本システムのフローチャート

6.3 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．

6.3.1 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．

6.3.2 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．

.

第 7 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 8 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')
```