

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 コーヒーの起源と伝搬の歴史 .....	2
2.3 コーヒーの流行について .....	3
2.3.1 ファーストウェーブ .....	3
2.3.2 セカンドウェーブ .....	3
2.3.3 サードウェーブ .....	4
第 3 章 抽出方法と抽出器具について	5
3.1 抽出方式 .....	5
3.1.1 浸漬式 .....	5
3.1.2 透過式 .....	5
3.1.3 高圧抽出式 .....	5
3.1.4 真空濾過式 .....	5
3.2 抽出器具 .....	6
3.2.1 コーヒープレス .....	6
3.2.2 プアオーバー .....	6
3.2.3 エスプレッソ .....	6
3.2.4 ソロフィルター .....	6
3.2.5 サイフォン .....	7
3.2.6 クレバードリッパー .....	7
3.2.7 エアロプレス .....	7
第 4 章 コーヒー抽出の学習方法	8
4.1 抽出の基本的な学習方法 .....	8
4.1.1 コーヒーの学習が可能な Web サイトについて .....	8
4.1.2 既存の音声認識可能な Web レシピについて .....	8
第 5 章 プログラミング言語と音声認識について	9
5.1 HTML・CSS とは .....	9
5.2 音声認識とは .....	9
5.2.1 音声認識の仕組み .....	9
5.2.2 WebSpeechAPI .....	10
5.2.3 音声認識を可能にする JavaScript について .....	10
5.2.4 音声認識によるページ移動 .....	10

第 6 章 本研究で開発する Web レシピの概要	11
6.1 コンセプトとページ構成について . . . . .	11
6.2 実装機能 . . . . .	11
6.2.1 抽出器具選択画面 . . . . .	12
6.2.2 レシピ閲覧画面 . . . . .	12
第 7 章 結言	14
第 8 章 参考文献	16
付録 A 作成したプログラム	17

## 図目次

2.2-1	UCC ホームページより:コーヒーの軌跡	3
6.2-1	抽出器具選択画面	12
6.2-2	レシピ閲覧画面	13

## 表目次

5.2-1 音声認識の方法論の変遷 (河原,2018) . . . . .	10
---------------------------------------	----

# 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。

世界的なコーヒーの流行は、ファーストウェーブと呼ばれる大量消費時代から始まり、その後、風味に重きが置かれたセカンドウェーブが到来する。そして近年では、高品質なコーヒーをより良い淹れ方で味わうサードウェーブコーヒーが流行している。

しかし、美味しいコーヒーを淹れるためには抽出器具に合わせたテクニックが必要である。抽出手順や、器具ごとの特徴の違いなどから学習の難易度が高くなり、コーヒーを淹れることの敷居も高くなっている。また、実際の抽出中は、両手が塞がっていたり手が濡れているなど、レシピブックを捲ることに対する障害も多い。クックパッドや macaroni のような、音声認識でレシピを閲覧できる Web サイトは複数存在するが、コーヒーの抽出に特化したサイトは無いことが問題点である。

そこで、様々な抽出器具のレシピが閲覧可能であり、音声認識によってページ移行が可能な Web レシピがあれば、問題点の改善に繋がると考えた。本研究では上記の Web レシピを作成することを目的とする。

## 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 コーヒーの起源と伝搬の歴史

コーヒーノキ（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。UCC ホームページより、コーヒーの軌跡を図 1 に示す。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のシェアの多くを占めている。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る。高品質なコーヒーを淹れる上で、アラビカ種を使用することはもはや前提といつても過言ではないシェアを誇るが、近年では、耐病性向上の観点からハイブリット種の開発も行われている。

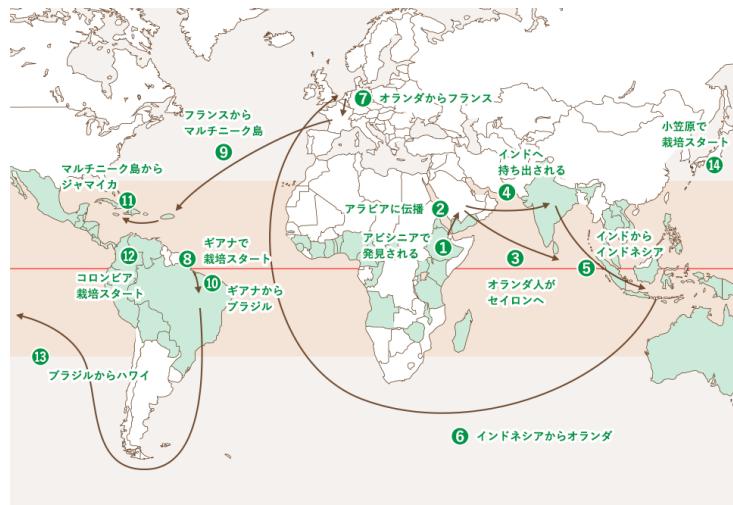


図 2.2-1: UCC ホームページより:コーヒーの軌跡

## 2.3 コーヒーの流行について

ここでは、コーヒーの流行を時代別に説明する際に使われる、ウェーブという表現について説明する。コーヒーはその時代ごとに求められる姿を変え、ファーストウェーブ（第一の波）、セカンドウェーブ（第二の波）、サードウェーブ（第三の波）へとシフトしていく。このウェーブという表現は、明確な時間の括りを示すものではないが、流行りの移り変わりとしてコーヒーを語る上で多く用いられる。

### 2.3.1 ファーストウェーブ

ファーストウェーブ（第一の波）の期間は、1800 年代後半～1960 年頃と言われている。この流行はコーヒーの大量消費時代であり、真空保存パックの発明や、インスタントコーヒーが発売された時代である。真空保存パックの発明は、コーヒーの保存期間を劇的に延長し、地方だけでなく大都市でも新鮮なコーヒーが飲めるようになった。インスタントコーヒーの発明は、コーヒーを楽しむことをより手軽な事とした、第一次・第二次世界大戦において多くの兵士が愛飲したと言われている。しかしその反面、この時代のコーヒーは風味がとても乏しく、粗悪品が多く出回る時代でもあった。

### 2.3.2 セカンドウェーブ

セカンドウェーブ（第二の波）の期間は、1960 年頃～2000 年頃と言われている。ファーストウェーブの際に、一般的かつ手軽な飲み物として普及が進んだコーヒーであったが、普及が進むとともに劣悪な品質が問題化した。そして、STARBUCKS などのシアトル発祥のカフェを代表とする、味にフォーカスした流行が起こる、そ

れがセカンドウェーブである。コーヒーそのものだけでなく、カフェラテなどのエスプレッソドリンクもシアトル系カフェの台頭とともに普及した。

### 2.3.3 サードウェーブ

サードウェーブ（第三の波）の期間は、2000年頃～現在も続くと言われている。セカンドウェーブの際に、品質についての意識が向上したことを皮切りに、高品質かつそのコーヒーの生まれや、農園、淹れ方、加工法などが重視される流行が訪れる事となる。それがサードウェーブコーヒーである。コーヒーの抽出技術を競う大会や、カップオブエクセレンスと呼ばれるその年のコーヒーで最高品質の物を決める評議会が開催されるなど、一杯の価値にフォーカスされている流行である。また、昔から日本のカフェで行われてきたハンドドリップは、日本でこそ当たり前の光景だったものの、海外では目新しいものであり、この流行とともに注目を集めていく事となる。

## 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

本節では、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、コーヒー豆をお湯や水に浸し、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。特に水出しの場合、お湯に比べ抽出スピードが遅いため、丸一日時間を要する場合もある。

#### 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。日本ではハンドドリップが、喫茶店で長い間提供されていたこともあり、認知度の特に高い抽出方式であるが、海外では比較的新しい抽出方法である。

#### 3.1.3 高圧抽出式

高圧抽出式は、お湯を浸したコーヒー豆に高い気圧をかけることにより、一気に味を抽出する方式である。エスプレッソマシンでは9気圧ほどになる場合もある。モカポッドやパーコレーターなども気圧を使用するが、こちらは1.5気圧ほどのため、高圧抽出とは呼ばれない場合が多い。

#### 3.1.4 真空濾過式

真空濾過式はサイフォンに代表される抽出方式であり、意図的に準真空状態を作り出すことにより、コーヒーを抽出する。この方式を採用している器具は少なく、日常で目にする器具はサイフォンがほとんどだが、同じく真空状態を意図的に利用する器具で、CLOVERと呼ばれる浸漬式と真空バキュームを組み合わせた器具も存在する。

## 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。紅茶でいうティーブレーカーのように、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いる。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 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。

### 3.2.4 ソロフィルター

ソロフィルターは、カフェ等ではありませんが、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ツ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。

### 3.2.5 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。

### 3.2.6 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。

### 3.2.7 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、シェアを順調に伸びている。

## 第 4 章 コーヒー抽出の学習方法

本章では、コーヒー抽出の基本的な学習方法を説明する。

### 4.1 抽出の基本的な学習方法

コーヒー抽出の学習方法は、他の学問と同様に様々な学習スタイルが存在する。近年では、多くの個人経営カフェや、チェーン店においてもコーヒーセミナーという形で、実践を伴った学習ができる。また個人学習においては、抽出に関することに主題を置いた書籍が多数出版されている他、インターネット上で抽出レシピやコツなどを簡単に閲覧することができる。

#### 4.1.1 コーヒーの学習が可能な Web サイトについて

インターネット上の Web サイトでは、コーヒーの基本知識から、実際の抽出方法に至るまで多くの情報が閲覧できる。個人で運営されている Web サイトだけでなく、コーヒー業界の大手である UCC や、STARBUCKS, TULLY'S などの企業も自社ページで、器具ごとの抽出手順や実際の抽出動画を公開している。

#### 4.1.2 既存の音声認識可能な Web レシピについて

既存の音声認識可能な Web レシピで特に有名なものとして、クックパッドが挙げられる。この Web レシピは、料理レシピの閲覧・投稿が可能であり、調理手順をステップごとに確認することができる。元々は音声認識は実装されていなかったが、Amazon が発売している、スマートスピーカー「Amazon Echo」のクックパッドスキルを使用することで、音声認識が使用できる。機能は主にレシピの検索や読み上げであり、画面ではなく音声を聞きながらの調理をすることになる。スマートフォンアプリで使用できるものとしては、macaroni という Web レシピが存在する。こちらは、調理動画を閲覧する際に音声認識が可能であり、「サイセイ」、「マキモドシ」などの発音で、動画の再生をコントロールできる。

## 第 5 章 プログラミング言語と音声認識について

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と音声認識について説明する。

### 5.1 HTML・CSS とは

HTML(Hyper Text Markup Language) とは、プログラミング言語の 1 つである。主に Web ページを作成する際に使用される言語である、単なる文字の集まりである文章に、見出し・ボタン・リストなどの役割を指定し、コンピュータに認識させることができる。現存するほとんどの Web ページが HTML によって構造を作られている。

CSS(Cascading Style Sheets) とは、HTML に記載された役割を持つ文章に対し、サイズ指定を行うことや、文字色の変更、配置を調整することができる言語である。HTML 単体の記述では、単調な文章表記やレイアウトになってしまふため、多くの Web サイトで CSS を用いたデザイン調整が行われている。使用方法は、HTML ファイルから別の CSS ファイルを読み込むというだけでなく、HTML ファイル中に直接 CSS を書くこともできるなど、用途に合わせた柔軟な記述ができるようになっている。

### 5.2 音声認識とは

音声認識とは、人間が発する声をコンピュータ上で解析を行い、テキストとして認識する技術である。

#### 5.2.1 音声認識の仕組み

音声認識の仕組みは利用するサービスによって異なるほか、その時代によっても音声認識の方法論が移り変わっている。河原（2018）の、音声認識の方法論の変遷を表として、表 reftb:onsei に示す。この表 1 を見ると現在の音声認識の方法論は 4.5 世代にあたり、ニューラルネットワークの利用が主流になっていることがわかる。ニューラルネットワークはいわば機械学習の類であり、それらを応用することで音声認識を成し遂げている。実際に、Google 社の手がける音声認識サービス「Google Cloud Speech API」においてもニューラルネットワークモデルが利用されていることが明記されている [1]。

表 5.2-1: 音声認識の方法論の変遷 (河原,2018)

第 1 世代	1950 ~ 1960 年代	ヒューリスティック
第 2 世代	1960 ~ 1980 年代	テンプレート (DP マッチング, オートマトン)
第 3 世代	1980 ~ 1990 年代	統計モデル (GMM-HMM, N-gram)
3.5 世代	1990 ~ 2000 年代	統計モデルの識別学習
第 4 世代	2010 年代	ニューラルネット (DNN-HMM, RNN)
4.5 世代	2015 年~	ニューラルネットによる End-to-End

## 5.2.2 WebSpeechAPI

WebSpeechAPI とは, JavaScriptAPI の 1 つである. JavaScript と Web ブラウザのみで, 音声の合成や音声認識を可能にする. 音声の合成は入力した文字を機械側に発音させる機能であり, 音声認識は音声をテキストとして入力する機能である, 本 Web アプリケーションの音声認識はこの WebSpeechAPI を使用している.

## 5.2.3 音声認識を可能にする JavaScript について

この文章必要だろうか?

## 5.2.4 音声認識によるページ移動

この文章必要だろうか?

## 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 コンセプトとページ構成について

コーヒーの抽出は目的に合わせ抽出器具を使い分けるため、それぞれの特徴を踏まえた上で器具の選定を行うことが重要である。そのため、本研究で開発した Web レシピのページは、「抽出器具選択画面」、「レシピ閲覧画面」の二段階で構成している。抽出器具選択画面において、利用者はそれぞれの器具の特徴や風味の違いを理解し、レシピ閲覧画面で実際の抽出をサポートする。

### 6.2 実装機能

本研究で実装する機能は、「コーヒーの専門性の高い Web レシピの機能」と「音声認識」である。「コーヒーの専門性の高い Web レシピの機能」は、コーヒーの抽出器具ごとの情報や、その組み立て方および抽出手順である。前節にて述べたとおり、抽出をする際に使用する器具の特徴をあらかじめ理解していることはとても重要である。抽出手順においても器具ごとに異なるため、そのステップごとに個別のページを用意し、順に解説を行う。

「音声認識」は、抽出ステップ間の移動を行う際に使用する。実際の抽出中は両手が塞がることが多く、ページ移動のボタンを押せないことが予期されるため、ボタンの他に音声認識でもページを移動できるようにする。

### 6.2.1 抽出器具選択画面

「抽出器具選択画面」では、レシピを閲覧する抽出器具を選択する。実際の抽出器具選択画面を図 6.2-1 に示す。

この画面では、抽出方式の異なる 3 つの代表的な抽出器具と、ソロフィルターを含めた 4 種類を選択できる。プアオーバー（透過式）、コーヒープレス（浸漬式）、エスプレッソ（高圧抽出式）は、日本で特に馴染みの深いコーヒーの抽出方法であり、様々なカフェで提供されていることから選択した。ソロフィルター（より手軽な入門向け）は、カフェ等ではありません提供されないが、コーヒーを初めて淹れる人でも簡単に抽出でき、入門用の器具の一つとして選定した。

「レシピを見る」を選択することで、「レシピ閲覧画面」へと移動する。

「レシピ閲覧画面」では、実際に器具ごとのコーヒー抽出レシピを閲覧できる。

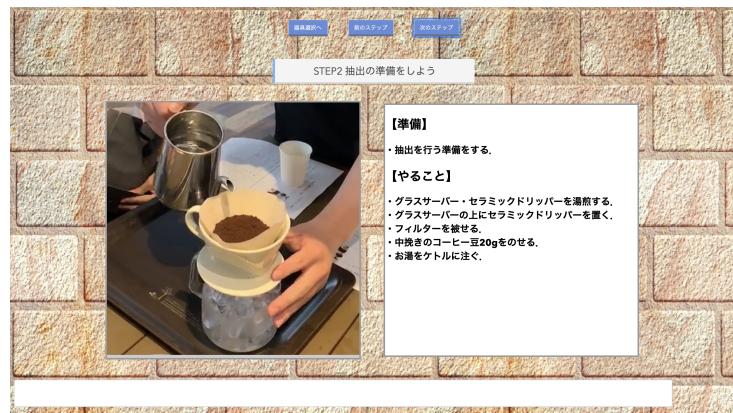


図 6.2-1: 抽出器具選択画面

### 6.2.2 レシピ閲覧画面

「レシピ閲覧画面」では、実際に器具ごとのコーヒー抽出レシピを閲覧できる。実際のレシピ閲覧画面を図 6.2-2 に示す。

ページは抽出ステップごとに分割している。1 つのステップ毎に、個別の画像と説明文を表示し、記載されている指示に従うことで、抽出を行うことができる。ステップ間は、画面上部のボタンを押すことで移動が可能であるほか、音声認識も実装している。画面下部の「音声認識を開始する」ボタンを押すことにより、音声認識処理が開始し、「次へ」・「戻る」の発音で、ステップ間の移動が可能である。



図 6.2-2: レシピ閲覧画面

## 第7章 結言：途中

コーヒーは、1章にて述べた通り、長い年月をかけて愛され、世界の人々を魅了してきた。また、その生産に当たっても、農家・バイヤー・焙煎士・加工者など多くの人々の手に渡りながら世界を巡り、最終的に消費者の元に訪れる。そのような長い過程を経たとしても、最終的には抽出次第で味は如何様にも変化する。難しいと感じる人も多いだろうが、そこがコーヒーの魅力でもあり、人々を惹きつける要因の1つだろう。

しかし、抽出手順や器具ごとの特徴の違いなどから学習の難易度が高くなり、コーヒーを淹れることの敷居も高くなっている実情は、その楽しみを阻害するだけでなく、新たなコーヒーの楽しみ方を模索する機会を失ってしまっていると考えられる。また、そこで本研究では、様々な抽出器具のレシピが閲覧可能であり、音声認識によってページ移行が可能なWebレシピを開発した。

今後このような、抽出アシストツールがより普及し、誰でも気軽にコーヒーを抽出できるようになることを期待している。

## 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] google , “Cloud Speech-to-Text”, <https://cloud.google.com/speech-to-text/?hl=ja>
- [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](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')
```