

麻雀点数計算システム 仕様書

1. 概要

本システムは、ユーザーがアップロードした麻雀のあがり形の画像から、物体検出技術（YOLO）を用いて麻雀牌を自動で認識し、役と点数を計算して表示するWebアプリケーションである。麻雀初心者の点数計算の補助や、対局記録のサポートを目的とする。

2. システム構成図

Role	Work	Target
User	画像と情報を入力	Frontend
Frontend	計算リクエスト	Backend
Backend	画像を渡して牌検出を依頼	YOLO
YOLO	牌の種類と位置情報を返す	Backend
Backend	牌の情報と補助情報を渡す	Calculator
Calculator	役・符・点数を返す	Backend
Backend	最終結果を送信	Frontend
Frontend	結果を画面に表示	User


3. 機能要件

機能分類	機能名	詳細
ユーザー機能	画像アップロード機能	<ul style="list-style-type: none">・ユーザーは手持ちのデバイスから、あがり形の画像ファイル(JPEG, PNG形式)をアップロードできる。・アップロードする画像の向きは問わない(システム側で補正を試みる)。
	補助情報入力機能	<ul style="list-style-type: none">・点数計算に必要な以下の補助情報を入力できる。<ul style="list-style-type: none">- 親 / 子- 場風 (東/南)- 自風 (東/南/西/北)- ドラ表示牌(システムでの自動検出も検討するが、確実性を高めるためユーザー入力も可能にする)- リーチ、一発、ツモ/ロン、嶺上開花、ハイテイ/ホウテイなどの状況
	結果表示機能	<ul style="list-style-type: none">・計算結果として以下の情報を画面に分かりやすく表示する。<ul style="list-style-type: none">- 認識された手牌(画像とテキスト)- 成立した役の一覧- 符、翻数- 最終的な点数(例:「子 満貫 8000点」)
システム機能	牌検出機能(YOLO)	<ul style="list-style-type: none">・アップロードされた画像から、YOLOモデルを用いて各麻雀牌を検出する。・検出対象: 萬子(1-9)、筒子(1-9)、索子(1-9)、字牌(東南西北白發中)。・赤ドラ(赤五萬、赤五筒、赤五索)を通常の牌と区別して検出できる必要がある。・検出結果として、各牌の「種類(ラベル)」と「画像内の位置(バウンディングボックス)」を取得する。





	手牌構成解析機能	<ul style="list-style-type: none"> ・検出された牌の位置情報と種類に基づき、手牌を整理する。 ・4面子1雀頭の形に分割する。 ・鳴き牌(ポン・チー・カン)と手牌を区別する。(※これは難易度が高い課題。牌の向きや配置のルールから推定する。将来的にはユーザーが修正できるUIも検討)
	点数計算機能	<ul style="list-style-type: none"> ・解析された手牌とユーザーが入力した補助情報に基づき、役を判定する。 ・符計算(副底、面子の構成、待ちの形、雀頭など)を行う。 ・翻数計算(成立役の翻数、ドラ、赤ドラ、裏ドラの合計)を行う。 ・計算した符と翻数から、点数表に基づいて最終的な点数を算する(満貫、跳満、倍満、三倍満、役満の切り上げ処理も含む)。

4. 画面仕様(ワイヤーフレーム)

4.1. 入力画面

- タイトル: 麻雀 点数計算AI 
- 画像アップロードエリア:
 - 「ファイルを選択」ボタンと、ドラッグ & ドロップで画像をアップロードできる領域。
 - 選択された画像のプレビューを表示。
- 補助情報入力フォーム:
 - 親/子: ラジオボタン (☐ 親 / ☒ 子)
 - 場風: ラジオボタン (☒ 東場 / ☐ 南場)
 - 自風: ラジオボタン (☒ 東 / ☐ 南 / ☐ 西 / ☐ 北)
 - ドラ表示牌: (任意) ユーザーが牌を選択入力できるUI。
 - 状況: チェックボックス (☐ リーチ / ☐ 一発 / ☐ ツモ ...など)
- アクションボタン:
 - 「計算する」ボタン

4.2. 結果表示画面

- 解析対象画像:
 - アップロードされた画像に、YOLOが検出した牌のバウンディングボックスを描画して表示。
- 計算結果:
 - 手牌:     (画像 or テキスト)
 - 成立役:
 - 役名1 (翻数)
 - 役名2 (翻数)
 - ドラ (翻数)
 - 符 / 翻数: 30符 / 4翻
 - 点数:
 - 子 満貫 8,000点
- アクションボタン:
 - 「もう一度試す」ボタン(入力画面に戻る)

5. 技術仕様

カテゴリ	技術・ライブラリ	バージョン/備考
プログラミング言語	Python	3.9 以上
Webフレームワーク	Flask	軽量で高速な開発に適しているため推奨 (Djangoも可)
物体検出	YOLO	YOLOv8 や YOLOv9 など、高性能な最新バージョンを推奨
AI/MLライブラリ	PyTorch	YOLOのバックエンドとして利用 (TensorFlowも可)
画像処理	OpenCV-Python, Pillow	画像の読み込み、リサイズ、バウンディングボックス描画など
数値計算	NumPy	検出結果の座標計算やデータ整形
フロントエンド	HTML5, CSS3, JavaScript	標準的なWeb技術

6. 開発ステップ(推奨)

1. 【フェーズ1】データセット構築とモデル学習
 1. 麻雀牌画像の収集: 様々な照明、角度、背景で撮影された麻雀牌の画像を大量に収集する。
 2. アノテーション: LabelImgなどのツールを使い、収集した画像内の各牌に「man1」「pin2」「aka_sou5」のようなラベルとバウンディングボックスを付与する。
 3. YOLOモデルの学習: 作成したデータセットでYOLOモデルを学習させ、精度の高い牌検出モデル(.pt ファイル)を作成する。
2. 【フェーズ2】コア機能の実装 (CUIベース)
 1. 学習済みモデルを使い、単一の画像から牌を検出・認識するPythonスクリプトを作成する。
 2. 認識結果(牌のリスト)から、役と点数を計算するロジックを実装する。まずはコマンドラインで完結するプログラムを目指す。
3. 【フェーズ3】Webアプリケーション化
 1. Flaskを用いて、画像アップロードを受け付けるAPIエンドポイントを作成する。
 2. アップロードされた画像を【フェーズ2】で作成したコア機能に渡し、計算結果をJSON形式で返すAPIを作成する。
 3. HTML/CSS/JavaScriptでユーザーインターフェースを構築し、APIと通信して結果を表示する。
4. 【フェーズ4】テストと改善
 1. 様々なパターンのあがり手画像でテストを行い、牌の誤認識や点数計算のバグを修正する。
 2. 特に、牌が重なっている場合や、照明が特殊な場合の認識精度を重点的に評価・改善する。
 3. (任意)Heroku, Render, Google Cloudなどにデプロイして公開する。

7. 課題と今後の拡張

- 牌の誤認識: 牌の重なり、光の反射、画像の不鮮明さによる誤認識が最大の課題となる。データセットの拡充とモデルの再学習が継続的に必要になる。
- 鳴き牌の判定: 牌の向き(横向き)や卓上の配置ルールから鳴き牌を推定する必要があり、100%の精度は難しい。将来的には、ユーザーが結果画面で鳴き牌を指定・修正できる機能を追加することが望ましい。
- 対応役の拡充: ローカル役など、特殊な役への対応。
- 動画対応: 対局動画からリアルタイムにあがり手を検出する機能。