特別研究報告題目

訓点資料の片仮名に関するデータベースシステムの開発

Development of a Database System
for Katakana Characters in Glossed Materials

主査　出口　利憲　教授
副査　田島　孝治　准教授

岐阜工業高等専門学校　専攻科　先端融合開発専攻

2023Y37　氏名　森　駿

令和 7 年（2025 年）1 月 31 日提出

Abstract

Glossed materials of classical Chinese texts have been an essential part of Japan's cultural heritage for centuries. The study of glosses in these materials has primarily focused on textual interpretations and historical language analysis. While various databases have been developed to digitize glossed materials, but no database has been specifically designed to analyze the katakana variations found in these sources. Since glossed materials from the Heian and Kamakura periods contain katakana characters, including variant forms, a database for these characters is essential for further research.

In this study, we developed a database system for katakana characters in glossed materials based on "尚書（古活字版第三種本）". The system was implemented using a PHP-based API server and a React(TypeScript) + Next.js + Material UI web application, allowing users to register, store, and view katakana characters along with their supplementary information. Additionally, we conducted an experiment utilizing the registered katakana images. Binarization using the Otsu method, feature extraction with EasyOCR, and cosine similarity measurements, I showed the feasibility of detecting similar characters in cases where noise was minimal.

目 次

第1章　序論

1.1　序論

古典中国語で記された漢籍や仏典は、日本において長い歴史を持つ文化的遺産であり、それらを母国語（日本語）で理解するための「訓読」は平安時代から長らく行われてきた。この訓読を補助するために、漢文の本文に付された記号や文字を「訓点」と呼び、現存する資料（訓点資料）には、訓点が加えられている。訓点資料の研究はこれまで、記述内容の解釈や正確な意味の把握に重きを置き、ヲコト点図や釈文の作成を通じた手法が確立されてきた。さらに、著名な訓点資料については現代語訳が作成され、これらの研究成果は書籍や Web 上で公開されている [1, 2, 3]。しかし、現代語訳は訳者の意図や他文献の影響を受けやすく、訓点資料そのものに対する分析や、日本語表現の歴史的変遷を研究するには適さないという課題がある。このような課題を背景に、近年では訓点資料の本文と訓点を直接扱える電子化データベースの構築が進められており、国立国語研究所所蔵の「尚書（古活字版第三種本）」については、すべてのヲコト点と本文中の漢字の電子化が行われ、データベースが構築されている [4, 5, 6, 7, 8]。一方で、訓点資料に含まれる片仮名に注目したデータベースは模写による「仮名字体表」のみで、一次資料（訓点資料）に基づくものは存在しない。しかし、平安・鎌倉時代の訓点資料には異体字を含む片仮名が使用されており、それらを分析することは訓点資料の研究をより深めるうえで不可欠である。本研究では、「尚書（古活字版第三種本）」に含まれる片仮名を対象に、片仮名に関するデータベースの設計および画像登録を行う。このデータベースによって一次資料に基づく片仮名研究を可能にし、訓点資料全体の解析の幅を広げることを目指す。また、登録された片仮名画像を活用し、データベース内の片仮名の誤登録の検出を試みることで、本研究の成果がデータベースの有用性向上に寄与することを評価する。

第2章　基本知識

2.1　漢文訓点資料 [9]

2.1.1　訓点資料

訓点資料とは、古典中国語で書かれた漢籍や仏典に対して、日本語での訓読を補助する目的で付与された符号（訓点）が記録された資料を指す。これらの資料は、日本の言語研究や歴史研究において貴重な一次資料として分析の対象となっている。また、その宗教的・文化的価値の高さから、寺院や文庫などで厳重に保管されているものも多い。

本研究で取り扱う訓点資料は、平安時代から鎌倉時代にかけて加点されたものであり、現代の高等学校漢文教科書に見られる句読点や返り点、送り仮名といった単純な記号体系とは異なる特徴をもつ記号が付加されている。これらの記号群を総称して「訓点」と呼び、訓読を正確かつ効率的に行うための重要な役割を果たしてきた。訓点資料は、単なる文献資料にとどまらず、当時の言語構造や社会的背景を反映した複合的な情報源として位置づけられる。

2.1.2　訓点

訓点とは、漢文を日本語として訓読する際に加えられた仮名や諸符号（ヲコト点、句読点、返り点など）の総称である。これらは、漢文本文の文法構造や語順を明確にするために記され、訓読を正確かつ効率的に行うための重要な役割を果たしている。

訓点は奈良時代の写経から始まり、平安時代、鎌倉時代にかけて訓点技術が発展してきた。具体的に使用される訓点の例として、以下のようなものが挙げられる。

- **句読点**: 文意の切れ目を示す記号。
- **科段点**: 段落や節の区切りを示す記号。
- **ヲコト点**: 訓読の語順を指示する記号。
- **声点**: 音読する際の声調や発音を示す記号。
- **語順符**: 語順を示すための補助記号。
- **合符**: 和訓や字音を示すための記号。
- **仮名点**: 送り仮名や振り仮名の記号。
- **漢文注**: 類音注や反切、注釈など、本文の補足説明として付けられる注記。

2.1.3　仮名文字

本研究における仮名文字とは、漢文資料に付与された訓点のうち、送り仮名や振り仮名として用いられている平仮名、片仮名などの文字を指す。これらは、日本語として漢文を訓読する際に必要不可欠な情報を補足し、文法的関係や発音を明確にする役割を果たしている。

2.2　訓点資料データベース

2.2.1　尚書（古活字版 第三種本）[8]

国立国語研究所が公開している「尚書（古活字版第三種本）」を対象とした詳細な分析に基づき、すべてのヲコト点を電子化して保存しているデータベースである。本データベースには、資料内の本文および割注の文字やヲコト点に関する情報が体系的に登録されている。

データベースは、以下のような構造を持つ複数のテーブルで構成されている。

- **charac**: 資料内の本文および割注の文字を登録。
- **elements**: 文字に加点されたヲコト点を登録。
- **gojunelements**: レ点や一二点などの語順点を登録。
- **kanaelements**: 文字に振られた仮名を登録。
- **place**: 各文字のページ内での論理的な位置を保存。
- **url**: 各ページの画像 URL を保存。

データベースに登録された情報は、検索ページを通じて文字の種類やヲコト点の種類、位置などの条件を指定して閲覧することが可能である。

2.2.2　先行研究による改修 [7]

従来の「尚書（古活字版第三種本）」を対象とした訓点資料データベースでは、文字に関連する検索結果として、該当する文字が存在するページ全体の画像とその出現位置の情報のみが表示されていた。そのため、文字同士を比較する場合には、ページ内から対象の文字を探し出し、複数の画面を見比べる必要があり、効率的な比較が困難であった。

また、検索機能においては、ヲコト点を座標や形状から指定して検索する方式が採用されていたが、これにはヲコト点に関する専門的な知識が必要であり、利用者が目的とするテキストを十分に理解していない場合には、実用性に限界があった。

これらの課題を解決するため、先行研究では資料内の全 286 ページを対象に文字位置の検出およびデータベースへの追加が行われた。ページ全体の画像から本文の文字を検出し、切り出した画像を以下の Table 2.1 のような構造のテーブルを用いてデータベースに登録

Table 2.1: Elements of new table.

Elements	Type
id_charac	varchar(12)
start_x	int(12)
start_y	int(12)
end_x	int(12)
end_y	int(12)
Width	int(10)
Height	int(10)
URL	char(255)

することで、検索ページで文字ごとの画像が直接表示できるようになり、ユーザが文字を容易に確認・比較できる環境が整備された。

本データベースでは、各文字の情報を以下の項目として管理している。

- **id_charac**: 既存の検索データベースで使用されている、各文字に割り当てられた固有 ID。
- **start_x**, **start_y**: 文字領域の始点を、それぞれ x 座標および y 座標で表す。
- **end_x**, **end_y**: 文字領域の終点を、それぞれ x 座標および y 座標で表す。
- **Width**, **Height**: 文字領域の幅と高さ。
- **URL**: IIIF Curation Viewer の部分矩形領域指定機能を用いて切り出した画像の URL。

これにより、検索結果から個別の文字画像へ直接アクセスできるようになり、従来のページ全体を表示する方式に比べ、より直感的な検索・比較が可能となった。

第3章　関連技術

3.1　フロントエンド

本研究では、アプリケーションを開発するにあたり、個人の環境に左右されることなく利用できるよう、ブラウザ上で動作する Web アプリケーションとして実装することにした。これにより、特定の OS やデバイスに依存せず、広範なユーザが容易にアクセスできる環境を構築できる。

フロントエンドの開発には、**React (TypeScript) + Next.js + Material UI** を採用した。これらの技術を選定した理由は、型安全性の確保、モダンな開発環境との適合性、およびコンポーネントベースの UI 設計を容易にするためである。本節では、それぞれの技術の概要と本研究における活用方法について述べる。

3.1.1　TypeScript[10]

TypeScript は、Microsoft によって開発された JavaScript のスーパーセットであり、静的型付けの概念を導入している。JavaScript の持つ柔軟な記述性を維持しつつ、型による安全性を強化することで、エラーの発生を抑制し、可読性と保守性を向上させることができる。

TypeScript の主な特徴として、以下の点が挙げられる。

- **静的型付け**: 変数や関数の型を事前に定義することで、実行前にエラーを検出できる。
- **インターフェースと型エイリアス**: 型の定義を明確にし、コードの構造を整理できる。
- **ECMAScript 準拠**: JavaScript の最新仕様をサポートし、トランスパイルを通じて広範な環境で動作可能。

本研究では、訓点資料データベースを扱うアプリケーションの実装において、文字、数値、ブーリアン、画像ファイルなど、多様なデータ型を取り扱う必要がある。そのため、開発の安定性を向上させ、意図しないデータ型の扱いによるバグを防ぐことを目的として、TypeScript を採用した。これにより、検索画面や切り出し画面の開発において、型の厳格な管理を行い、より安全で信頼性の高いデータ処理が可能となる。

3.1.2 React[11]

React は、Facebook（現 Meta）によって開発された JavaScript ライブラリであり、コンポーネントベースのユーザインターフェース（UI）開発を効率化するための機能を提供する。その柔軟性と高いパフォーマンスから、モダンなフロントエンド開発における標準的な選択肢となっている。

React の主な特徴として、以下の点が挙げられる。

- **コンポーネントベース**: UI を独立した再利用可能なコンポーネントとして構築できる。
- **仮想 DOM**: 実際の DOM に変更を加える前に仮想 DOM 上で差分を計算し、高速なレンダリングを実現する。
- **状態管理**: ‘useState‘ や ‘useReducer‘ などのフックを活用して、コンポーネント間の状態を簡潔に管理できる。

本研究では、React を基盤とした Next.js を採用することで、訓点資料データベースの検索画面や画像ビューアをモジュール化し、保守性や拡張性の高い設計を実現した。特に、コンポーネントベースの設計により、各 UI 要素を独立して管理・再利用できる環境を構築した。

3.1.3 Next.js[12]

本研究では、開発におけるフレームワークとして Next.js を採用した。Next.js は、React を基盤としたモダンな Web アプリケーション開発に必要な機能を提供するフレームワークであり、開発者体験（DX）の向上やパフォーマンスの最適化に適している。

Next.js の主な特徴として、以下の点が挙げられる。

- **サーバサイドレンダリング（SSR）**: サーバ側で HTML を生成することで、初期表示速度を向上させる。
- **静的サイト生成（SSG）**: ビルド時に静的 HTML を生成することで、高速かつスケーラブルなアプリケーションを提供。
- **ファイルシステムルーティング**: ページをファイルとして作成するだけで、ルーティングが自動的に設定され、直感的にページを構築できる。

3.1.4 MUI[13]

MUI は、Google の Material Design に基づいて構築された React コンポーネントライブラリであり、モダンなデザインと高いカスタマイズ性を兼ね備えている。特に、一貫性のある UI コンポーネントを迅速に構築するためのツールとして広く利用されている。

MUI の主な特徴として、以下の点が挙げられる。

- **豊富なコンポーネント**: ボタン、テーブル、モーダルなど、よく使われる UI コンポーネントが多数用意されている。
- **テーマのカスタマイズ**: デフォルトの Material Design に基づくテーマをベースに、色やフォント、スタイルを柔軟に変更可能。
- **アクセシビリティ対応**: 各コンポーネントがアクセシビリティ基準に準拠しており、全てのユーザにとって使いやすい設計が可能。

本研究では、MUI を利用して、訓点資料データベースの検索画面や切り出し画面などの UI を構築した。MUI のコンポーネントを活用することで、一貫性のあるデザインを短期間で実現し、テーマのカスタマイズ機能を用いて、本研究の目的に合った独自のデザインを適用することができた。また、アクセシビリティ対応が容易であることから、ユーザ全体にとって利便性の高いインターフェースを提供することが可能となった。

3.2 バックエンド

本研究のバックエンド開発には、オープンソースのソフトウェア群で構成される LAMP スタック（Linux, Apache, MySQL, PHP）を採用した。LAMP スタックは、ウェブサイトやウェブアプリケーションの構築に広く利用されており、その信頼性と柔軟性から多くの開発者に支持されている [14]。

LAMP スタックを選択した主な理由は以下に示す。

- **コスト効率**: すべてのコンポーネントがオープンソースであり、ライセンス料が不要なため、開発コストを抑えることができる。
- **実績と信頼性**: 長年にわたり多くのウェブアプリケーションで採用されてきた実績があり、その信頼性は高く評価されている。
- **コミュニティサポート**: 広範なコミュニティによるサポートが充実しており、問題解決や情報収集が容易である。
- **柔軟性**: 各コンポーネントの組み合わせや設定をプロジェクトの要件に応じてカスタマイズできる柔軟性がある。

以下に、LAMP スタックを構成する各要素について詳述する。

3.2.1 Ubuntu[15]

本研究では、バックエンドサーバのオペレーティングシステムとして Ubuntu 22.04 LTS を採用した。Ubuntu は、Canonical 社とコミュニティによって開発・維持されているオープンソースの Linux ディストリビューションであり、デスクトップ、サーバ、クラウド、および IoT デバイス向けに提供されている。

Ubuntu の主な特徴として、以下の点が挙げられる。

- **安定性と信頼性**: Ubuntu は長期サポート（LTS）バージョンを提供しており、最大 5 年間のセキュリティ更新とサポートを受けることができる。これにより、システムの安定稼働が保証される。
- **パッケージ管理の容易さ**: APT（Advanced Package Tool）を使用したパッケージ管理により、ソフトウェアのインストール、更新、削除が容易であり、システムの保守性が向上する。
- **コミュニティとサポート**: Ubuntu は広範なユーザコミュニティを持ち、豊富なドキュメンテーションやフォーラムが存在する。これにより、問題解決や情報収集が容易である。
- **セキュリティ機能**: デフォルトで多くのネットワークポートが閉じられており、UFW（Uncomplicated Firewall）などのツールを使用してファイアウォールの設定が簡単に行える。

3.2.2 Apache[16]

Apache（Apache HTTP Server）は、Apache ソフトウェア財団によって開発・維持されているオープンソースの Web サーバであり、UNIX や Windows を含む多様なオペレーティングシステム上で動作する。

Apache は、モジュール化された設計により、機能の拡張やカスタマイズが容易であり、認証スキームの追加や、Perl、Python、PHP などのサーバサイドプログラミング言語がサポートされている。

本研究では、Apache をバックエンドサーバとして採用し、PHP との連携により、動的な Web ページの提供や API エンドポイントの構築を行った。

3.2.3 MariaDB[17]

MariaDB は、MySQL から派生したオープンソースのリレーショナルデータベース管理システム（RDBMS）であり、MySQL の創始者である Michael Monty Widenius 氏によって開発が進められている。MariaDB は、MySQL との高い互換性を維持しつつ、性能や機能の向上が図られている。本研究では先行研究で用いられていたデータベースの環境に合わせ、MariaDB を採用した。

3.2.4 PHP[18]

PHP（Hypertext Preprocessor）は、動的な Web ページや Web アプリケーションの開発に広く利用されているサーバサイドのスクリプト言語である。そのシンプルさと柔軟性から、多くの開発者に支持されている。

PHP の主な特徴として、以下の点が挙げられる。

- **動的なコンテンツ生成**: PHP は HTML に埋め込む形で記述でき、ユーザの入力やデータベースの情報に基づいて動的にコンテンツを生成することが可能である。
- **豊富なライブラリとフレームワーク**: PHP には多彩なライブラリやフレームワークが存在し、開発効率を高めることができる。
- **データベースとの連携**: MySQL や PostgreSQL など、主要なデータベースと容易に連携でき、データの操作や管理が容易である。
- **コミュニティの活発さ**: PHP は広範なユーザコミュニティを持ち、公式ドキュメントや多様なリソースが提供されている。

本研究では、PHP8.3 をバックエンドの主要なプログラミング言語として採用し、データベースとの連携や API の構築を行った。

3.3 画像処理

3.3.1 大津の方法 [19]

大津の方法（Otsu's Method）は、画像の二値化において最適な閾値を自動的に求める手法の一つであり、判別分析法とも呼ばれる。画像ごとに適切な閾値を決定できるため、前処理として広く利用されている。この手法では、ヒストグラムのクラス内分離度が最大になる閾値を選択することで、二値化の最適な閾値を決定する。具体的には、ヒストグラムを左右に分割し、それぞれをクラス 1 とクラス 2 に分類する。全ての画素値について、この閾値を基準とした場合の分離度を計算し、分離度が最大となる画素値を閾値として選択する。クラス 1 の分散を σ_1^2、平均を μ_1、画素数を n_1、クラス 2 の分散を σ_2^2、平均を μ_2、画素数を n_2 とする。全体の平均を μ_0 とし、クラス内分散 σ_w^2 とクラス間分散 σ_b^2 は以下の式 (3.1)、(3.2) によって求められる。

$$\sigma_w^2 = \frac{n_1\sigma_1^2 + n_2\sigma_2^2}{n_1 + n_2} \tag{3.1}$$

$$\sigma_b^2 = \frac{n_1(\mu_1 - \mu_0)^2 + n_2(\mu_2 - \mu_0)^2}{n_1 + n_2} \tag{3.2}$$

この分散値を用いて、分離度 J は以下の式 (3.3) で計算される。

$$J = \frac{\sigma_b^2}{\sigma_w^2} \tag{3.3}$$

本研究では、大津の方法を用いることで、切り出した片仮名の画像を二値化し、画像処理の前処理として活用した。これにより、ノイズを抑えつつ、より鮮明な文字領域を抽出することが可能となった。

3.3.2 EasyOCR[20]

EasyOCR は、オープンソースの光学文字認識（OCR）ライブラリであり、Python を用いて簡単にテキストを抽出できるツールである。複数の言語に対応しており、ディープラーニングを活用して、高精度な OCR を実現している。EasyOCR は、日本語や中国語、英語など 80 以上の言語をサポートしており、多言語の文字認識が可能である。OCR の処理には、畳み込みニューラルネットワーク（CNN）とリカレントニューラルネットワーク（RNN）を組み合わせたモデルを採用しており、手書き文字や印刷文字の認識精度が向上している。また、シンプルな API を提供しており、数行のコードで画像からテキストを抽出できるため、他のアプリケーションとの統合も容易である。本研究で扱う片仮名は異体字を含むため、EasyOCR は OCR 目的ではなく、画像の特徴量を抽出する手法として利用した。具体的には、大津の方法によって二値化した片仮名画像を EasyOCR のモデルに入力し、中間層の出力として得られる、文字の形状や線のパターンを数値データとして畳み込んだものを取得した。この特徴量は、後述するコサイン類似度を用いた文字の比較において、類似度計算の基礎データとして活用される。

3.3.3 コサイン類似度 [21]

コサイン類似度（Cosine Similarity）は、2 つのベクトル間の角度の余弦を計算することで、ベクトルの類似度を測る手法である。主に文書検索や自然言語処理の分野で使用されるが、画像の特徴量ベクトルの比較にも応用できる。類似度の値は -1 から 1 の範囲をとり、1 に近いほどベクトル同士の方向が一致し、類似度が高いことを示す。コサイン類似度は、2 つのベクトル $\mathbf{A}$、$\mathbf{B}$ に対して、以下の式 (3.4) によって計算される。

$$\textbf{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{3.4}$$

ここで、$\mathbf{A} \cdot \mathbf{B}$ はベクトルの内積、$\|\mathbf{A}\|$ および $\|\mathbf{B}\|$ はそれぞれのベクトルのノルム（大きさ）である。

本研究では、EasyOCR によって抽出した片仮名画像の特徴量をベクトル形式に変換し、類似文字間の距離を測定するためにコサイン類似度を用いた。これにより、画像内の文字の類似度を数値的に評価することが可能となり、データベース内の誤りを検出する際の基準として活用した。

第4章　開発

4.1　データベース

従来のデータベースでは、漢字の語句に振られた一連の片仮名を文字列単位で保存し、情報を管理していた。そのため、Table 4.1 に示す「kanaelements」テーブルが存在している。各要素の意味は以下の通りである。

- **id_kanaelements**: 一連の片仮名文字列の ID
- **targetLength**: 片仮名が振られている漢字の文字数
- **position**: 片仮名が漢字の左右どちらにあるか
- **positionText**: 片仮名の位置を示すテキスト表現
- **style**: 墨や朱などの書式情報
- **text**: 振られている片仮名の文字列
- **id_charac**: 片仮名が振られている漢字の ID

本研究では、片仮名 1 文字ごとの情報をより詳細に管理するため、新たに「kana_charac」テーブルおよび「kana_notes」テーブルを追加した。

kana_charac テーブル

従来のデータベースでは、片仮名の語を文字列単位で保存していた。しかし、片仮名の個別分析や座標情報の管理を可能にするため、本研究では「kana_charac」テーブルを設計した。Table 4.2 にこのテーブルの構成を示す。

このテーブルでは、各片仮名を一意に識別するために「kana_charac_id」を主キーとし、以下の要素を定義している。

- **kana_charac_id**: 片仮名の ID
- **id_kanaelements**: 片仮名の属する語の ID（kanaelements テーブル）
- **character_index**: 片仮名の語における文字の順番
- **charac**: 片仮名の文字

Table 4.1: Elements of kanaelements table.

Elements	Type
id_kanaelements	int(5)
targetLength	int(1)
position	int(1)
positionText	varchar(3)
style	varchar(1)
text	varchar(12)
id_charac	varchar(12)

- **x1, y1, x2, y2**: 資料の全体画像における片仮名の座標（2 点で囲まれる長方形領域）

この構造により、各片仮名の個別情報と座標を統一的に管理できるようになり、文字比較や類似文字検索の実装に貢献する。

kana_notes テーブル

片仮名に関する画像を登録する際、資料の汚れやイレギュラーな表記などを補足する情報が必要となる。これらの情報を管理するために、「kana_notes」テーブルを追加した。Table 4.3 にこのテーブルの構成を示す。

このテーブルの主な要素は以下の通りである。

- **id_kanaelements**: 片仮名の語単位の ID（kanaelements テーブルとの関連）
- **note**: 片仮名に関する補足情報

本研究では、既存の ‘kanaelements‘ テーブルを拡張し、新たに ‘kana_charac‘ テーブルおよび ‘kana_notes‘ テーブルを導入することで、片仮名の詳細なデータ管理と文字単位での解析を可能にした。

Table 4.2: Elements of charac_kana table.

Elements	Type
kana_charac_id	int(11)
id_kanaelements	int(11)
character_index	tinyint(4)
charac	varchar(1)
x1	double
x2	double
y1	double
y2	double

Table 4.3: Elements of kana_notes table.

Elements	Type
id_kanaelements	int(11)
note	text

4.2 APIサーバ

4.2.1 基本設計

本研究では、訓点資料のデータ管理を効率化するため、API サーバを設計・実装した。API サーバは、フロントエンドの Web アプリケーションとデータベースを接続し、データの取得・登録・更新を担う。

API サーバの実装には、**Ubuntu + Apache + PHP** を採用した。また、データベースには **MariaDB** を使用し、バックエンドでのデータ管理を行う。

本 API サーバは、以下の主要エンドポイントを提供する。

- **ユーザ認証 API**: JWT (JSON Web Token) を用いた認証機能
- **ページ関連 API**: 訓点資料や片仮名データの取得
- **画像関連 API**: 片仮名画像の登録・取得

API の基本仕様として、すべてのパラメーターは JSON 形式で送受信を行う。エラーが発生した場合、サーバは 200 番台以外のステータスコードを返却し、レスポンス JSON 内の error フィールドにエラーメッセージを格納する。

また、認証関連 API を除くすべてのエンドポイントでは、アクセス認証を必須とする。認証には Authentication ヘッダーに「Bearer アクセストークン」を付与する方式を採用し、アクセストークンの有効期限は発行から 24 時間とする。有効期限の管理は JWT の payload 内に含まれる exp フィールドによって行われ、クライアント側で適宜確認可能である。

4.2.2 ユーザ認証

本研究では、**JWT (JSON Web Token)** を用いた認証機構を実装した。ユーザはアカウントを作成し、ログインすることで、各 API にアクセス可能な認証トークンを取得する。

アカウントの作成

ユーザが新規アカウントを作成するための API を提供する。リクエスト時に必要な情報を送信すると、サーバ側で検証を行い、成功した場合に認証トークンを返却する。 ID とハッシュ化した password はデータベースに保存される。今回、登録用コードは事前に発行したものを用いるように実装した。

- **エンドポイント**: /auth/register.php
- **メソッド**: POST

- **パラメータ**:
  - **id**: ユーザの識別 ID
  - **password**: パスワード
  - **code**: 登録用コード
- **レスポンス**:
  - **token**: 認証用トークン (JWT)

ログイン

既存のユーザが API にアクセスするためにログインし、認証トークンを取得する。リクエスト時に ID とパスワードを送信し、認証が成功した場合にトークンを発行する。

- **エンドポイント**: /auth/login.php
- **メソッド**: POST
- **パラメータ**:
  - **id**: ユーザの識別 ID
  - **password**: パスワード
- **レスポンス**:
  - **token**: 認証用トークン (JWT)

認証フロー

本 API では、JWT を用いた方式でユーザ認証を行う。ユーザがログイン後に取得するトークンは **Authentication ヘッダー** に付与し、各 API へリクエストを送信する。

- 認証トークンの有効期限は発行から 24 時間
- トークンの有効期限は JWT の payload 内の exp フィールドで確認可能
- 認証関連 API 以外のすべてのエンドポイントでは、リクエストヘッダーに「Bearer アクセストークン」を指定して認証が必要

4.2.3 ページ関連 API

本 API では、訓点資料の各ページデータを取得・管理するためのエンドポイントを提供する。主に、ページの一覧表示、ページ内の文字情報取得、メモの検索・登録といった機能を実装している。

ページ一覧表示

このエンドポイントでは、データベース内から資料の全ページの一覧を取得し、ページ ID、巻・ページ情報、ページ参照キーを提供する。この情報を用いて扱うページを選択し、後述のエンドポイントで詳細なデータをリクエストする。

- **エンドポイント**: /page/list.php
- **メソッド**: GET
- **パラメータ**: なし
- **レスポンス**:
  - **pages**: ページ情報の配列
    1. **id**: ページ ID
    2. **page**: ページ名（例: 巻 X：Y、X →巻 / Y →ページ）
    3. **key**: ページ参照キー

ページ内文字一覧

特定のページに含まれる漢字やカナ情報を取得するためのエンドポイントである。このエンドポイントでは、指定されたページの漢字一覧を取得し、それぞれの漢字に対応するカナ情報も提供する。これをもとに特定の片仮名を選択し、後述の画像のリクエストや座標の登録を行う。

- **エンドポイント**: /page/text.php
- **メソッド**: GET
- **パラメータ**:
  - **key**: ページ参照キー
- **レスポンス**:
  - **characters**: ページ内の漢字情報の配列
    1. **id**: 文字 ID
    2. **line**: 行名
    3. **character**: 文字
    4. **warityu**: 割注情報
    5. **warityu_kaigyo**: 割注の改行情報
    6. **kana**: カナ情報の配列
       - **id**: カナ ID
       - **text**: カナの文字列

- **note**: メモ情報
- **position**: 位置情報
  (a) **number**: 0 (右) / 1 (左)
  (b) **text**: 位置情報の文字列
- **length**: 漢字の文字長
- **characters**: カナ文字情報の配列
  (a) **id**: カナ文字 ID
  (b) **character**: 文字
  (c) **index**: カナ内の何文字目か
  (d) **position**: 切り抜き場所情報
    - **x**: X 座標 (配列) または NULL
    - **y**: Y 座標 (配列) または NULL

メモ検索

カナ情報に紐づくメモを検索するためのエンドポイントである。特定の片仮名文字列に関連するメモを取得するほか、ID を指定しない場合はすべてのメモを取得できる。

- **エンドポイント**: /page/note.php
- **メソッド**: GET
- **パラメータ**:
  - **id**: カナ ID（任意）
- **レスポンス**:
  - **notes**: メモ情報の配列
    1. **id**: カナ ID
    2. **text**: 文字
    3. **note**: メモ

メモ登録

カナ情報に紐づくメモを新規登録するためのエンドポイントである。特定の片仮名文字列を指定し、メモ内容を送信すると、該当するカナに対してメモが登録される。

- **エンドポイント**: /page/note.php
- **メソッド**: POST
- **パラメータ**:

- **id**: カナ ID
- **note**: メモの内容

- **レスポンス**: なし

4.2.4 画像関連 API

元画像取得

元画像の取得エンドポイントでは、資料のページ画像を取得する。取得した画像は、後の画像処理や切り取りの基準として使用される。

- **エンドポイント**: /image/original.php
- **メソッド**: GET
- **パラメータ**:
  - **id**: ページ参照キー
  - **token**: アクセストークン（ヘッダーにトークンを設定できない場合に使用）
- **レスポンス**: 画像ファイル（image/jpeg）

切り取り用画像取得

このエンドポイントでは、特定のカナ文字を切り取るための画像を取得する。クライアント側で適切な座標を取得し、切り取り位置登録 API に送信する。

- **エンドポイント**: /image/cut.php
- **メソッド**: GET
- **パラメータ**:
  - **id**: カナ文字 ID
  - **area**: 切り取り範囲
    - 0 以上の整数で指定。通常は 1 を指定
    - 切り取り範囲がずれている場合、2 以上を指定すると範囲の拡大が可能
  - **token**: アクセストークン（ヘッダーにトークンを設定できない場合に使用）
- **レスポンス**: 画像ファイル（image/jpeg）

切り取り位置登録

切り取り位置をサーバに登録し、カナ文字の座標情報をデータベースに保存する。登録された座標を基に、切り取り済み画像を取得できるようになる。

- **エンドポイント**: /image/cut.php
- **メソッド**: POST
- **パラメータ**:
  - **id**: カナ文字 ID
  - **area**: 切り取り範囲
    - 0 以上の整数で指定。通常は 1 を指定。
    - 切り取り用画像取得に使用した値と同じ値を指定。
  - **position**: 切り取り位置（相対座標）
    - ‘[x1, y1, x2, y2]‘ の形式で指定。
    - 切り取り用画像取得 API から得た座標情報を送信。
    - サーバ側で全体画像の座標に変換して登録。
- **レスポンス**: なし

切り取り済み画像取得

切り取り位置が登録されたカナ文字の画像を取得するエンドポイントである。登録されていない場合、404 エラーが返される。

- **エンドポイント**: /image/character.php
- **メソッド**: GET
- **パラメータ**:
- **id**: カナ文字 ID
  - **レスポンス**: 画像ファイル（image/jpeg）
    - 切り取り位置が登録されていない場合は 404 エラー。

4.3 Web アプリケーション

4.3.1 基本設計

本研究では、訓点資料の効率的な管理と操作を可能にする Web アプリケーションを設計・実装した。本アプリケーションは、以下の技術スタックを基盤として構築されている。

- **React (TypeScript)**:
  - コンポーネントベースの UI 設計を採用し、コードの再利用性を向上。
  - 型安全性を確保するため、TypeScript を併用。
- **Next.js**:
  - サーバサイドレンダリング (SSR) により、初期表示速度を向上。
  - 静的サイト生成 (SSG) を利用し、高速・スケーラブル化。
  - ファイルシステムルーティングにより、直感的かつ簡潔なページ構成を実現。
- **Material UI**:
  - Material Design に基づいたコンポーネントで一貫性のある UI を構築。
  - アクセシビリティ基準を満たし、ユーザに配慮したデザインを提供。

このアプリケーションでは、API を通じてバックエンドと通信し、以下の主要機能を提供する。

4.3.2 ユーザ認証

先述した通り、本アプリケーションでは、JWT (JSON Web Token) を用いてユーザ認証を行う。認証機能を備えることで、特定のユーザのみがデータを操作できるようになり、適切なアクセス制御が可能となる。

認証画面の設計

本アプリケーションでは、ユーザのログインおよびアカウント登録を行うための認証画面を用意した。Figure 4.1 に示すように、ログイン画面では ID と Password を入力し、認証に成功するとアクセストークンが発行される。また、アカウントを持たないユーザ向けに、Figure 4.2 に示す Sign Up 画面を用意し、ID・パスワード・登録コード (Keycode) を入力することでアカウント作成が行える。

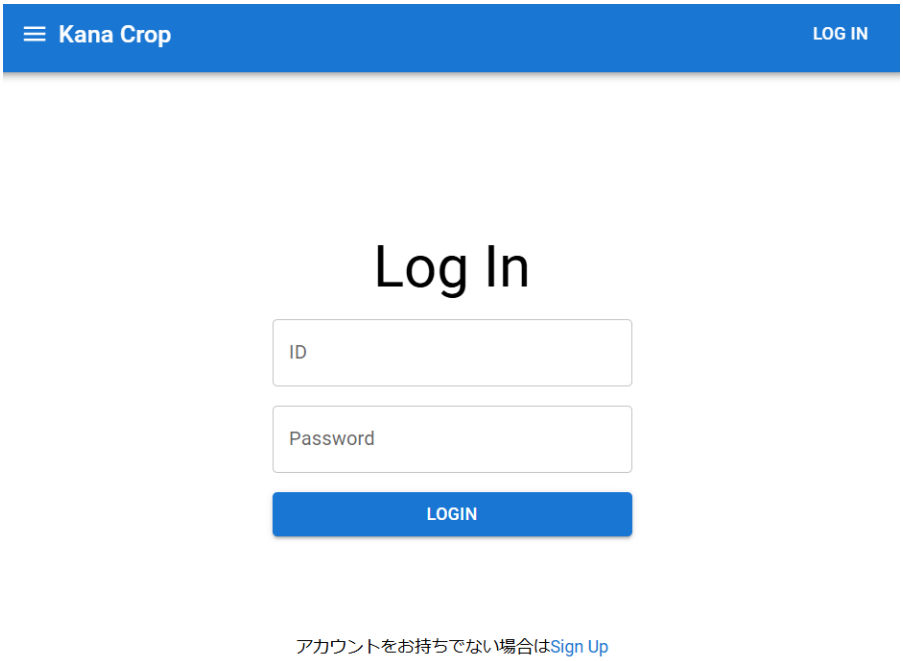

Figure 4.1: Login screen.

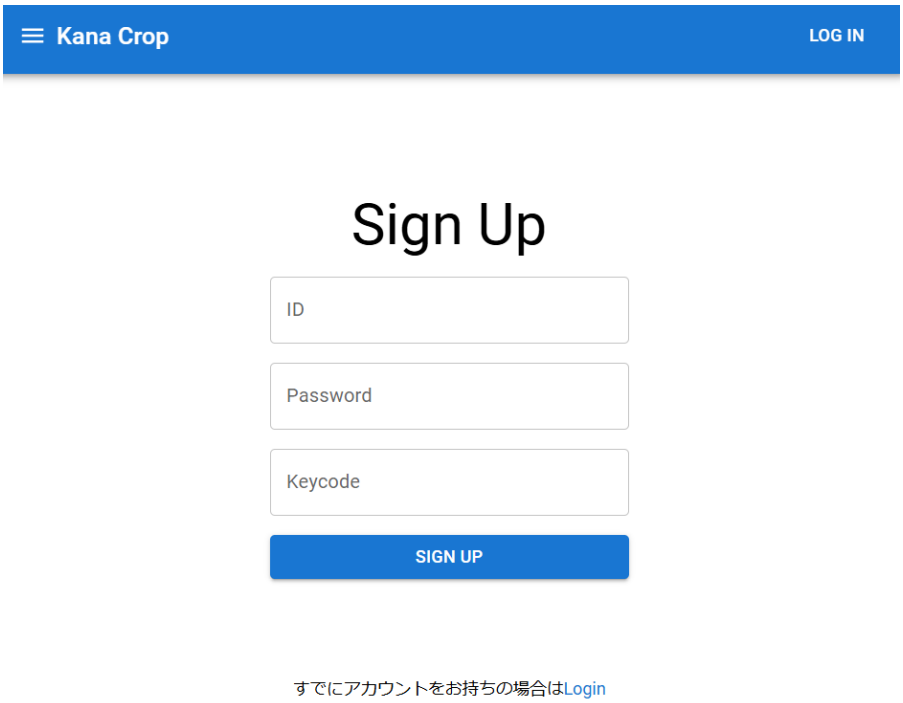

Figure 4.2: Account registration screen.

フロントエンドでの認証処理

本システムでは、認証情報の管理を JWT + LocalStorage で行う。具体的な処理の流れは以下の通りである。

1. ユーザが ID と Password を入力し、ログインリクエストを送信する。
2. API サーバ (/auth/login.php) で認証が成功すると、JWT が発行される。
3. フロントエンドでは受け取った JWT を localStorage に保存する。
4. 画像の閲覧・登録ページでは、localStorage に保存された JWT を参照し、認証情報を検証する。
5. JWT が存在しない場合、ログインページ (/login) へリダイレクトする。

トークンの管理とページ遷移

本アプリケーションでは、'localStorage' に JWT を保存し、ユーザがログイン状態を維持できるようにしている。トークンの管理とページ遷移は以下のように設計した。

- 認証が必要なページでは、トークンが無い場合ログインページへリダイレクトする。
- ログイン時に JWT を localStorage に保存し、次回アクセス時にセッションを維持する。
- ログアウト時には localStorage をクリアし、ログインページへ遷移する。

4.3.3 片仮名画像の切り出し機能

本システムでは、訓点資料内の片仮名を正確にデータ化するため、片仮名の画像をユーザが手動で切り出す機能を提供する。Figure **??** のページでユーザがセレクターで特定の片仮名を選択すると、Figure 4.4 のように画像が表示され、その画像内の領域を指定し、データベースに登録できる仕組みを構築した。

セレクターの構成

カナセレクターは、以下の選択フィールドで構成される。

- **Page（ページ）**：資料内の対象ページを選択する。
- **Text（文字）**：選択したページ内の特定の漢字を指定する。
- **Kanas（片仮名の語）**：漢字に付随する片仮名の語（複数文字）を選択する。
- **Kana（片仮名）**：上記の語の中から、個別の片仮名を選択する。
- **Area（切り取り範囲のスケール）**：切り取り用画像のスケールを指定する。

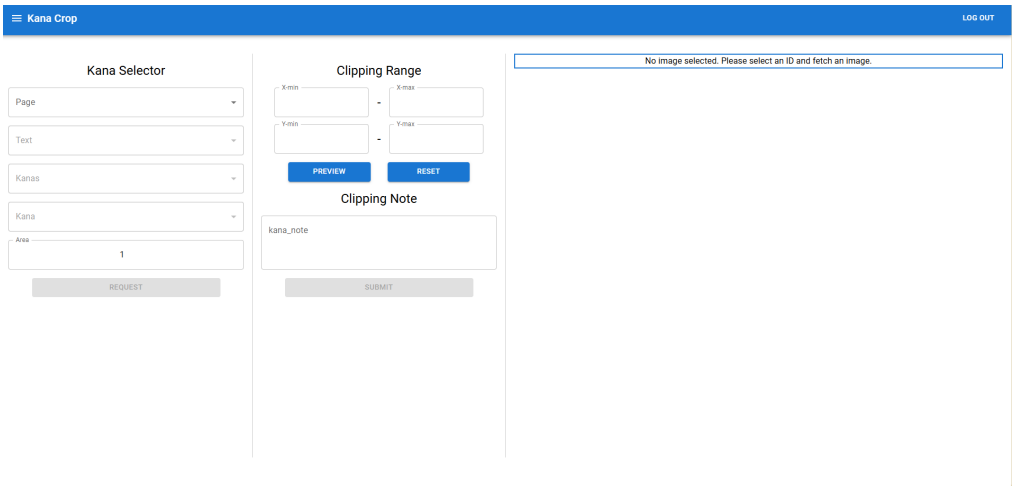

Figure 4.3: Katakana image cropping screen (initial state).

画像の切り出し手順

1. **対象の片仮名を選択する**
   - セレクターを用いて、処理対象の片仮名を選択する。
2. **画像をリクエスト**
   - 選択した片仮名に対応する画像が、右側のビューアに表示される。
3. **切り取り範囲の選択**
   - 画像をクリックし、切り取りたい範囲を囲むように 2 点を指定する。
   - 領域選択が完了すると、青色の枠が表示され、対象範囲が明示される。
4. **切り取りプレビュー**
   - 「PREVIEW」ボタンを押すことで、選択範囲の切り出し結果を確認できる。
5. **リセット**
   - 「RESET」ボタンを押すことで、選択した座標をクリアし、再度範囲指定をやり直す。
6. **メモの登録**
   - 必要に応じて、「Clipping Note」欄に切り出した片仮名に関するメモを入力する。
   - 例えば、「汚れがある」「登録と異なる文字の可能性がある」など、後の解析に役立つ情報を記録できる。

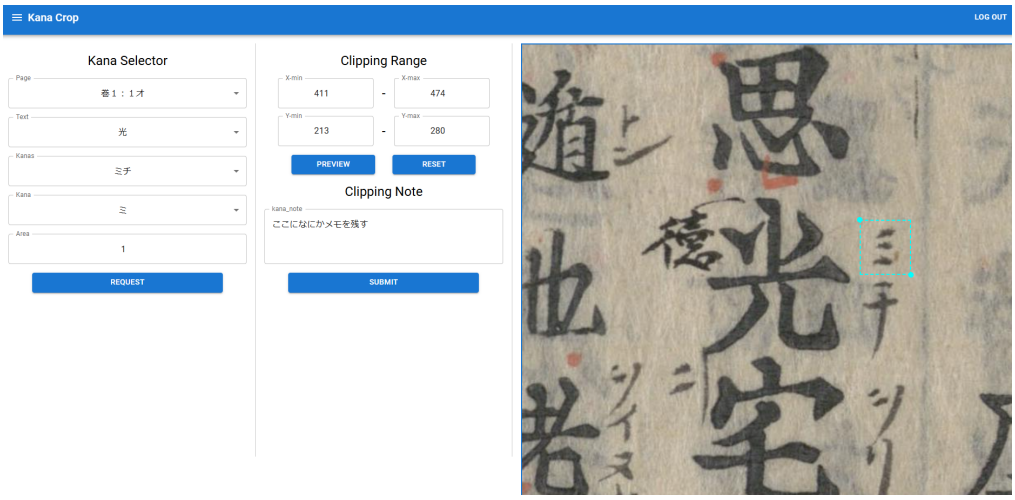

Figure 4.4: Katakana image cropping screen (cropping state).

7. **データ送信**

   - 「SUBMIT」ボタンを押すことで、切り出した画像の座標情報とメモをサーバに登録する。

データの管理

切り取った片仮名の座標情報は、データベース内の kana_charac テーブルに登録される。登録時には、以下の情報が適した API のエンドポイントを通して送信する。

- **id**: 片仮名の一意の識別 ID
- **area**: 切り取り範囲のスケール
- **position**: 切り取った領域の座標 [x1, y1, x2, y2]
- **note**: 登録時のメモ情報

本機能を導入することで、ユーザは精度の高い片仮名データの切り出しを行い、資料のデジタル化に貢献できる。また、直感的な UI 操作により、専門知識がない利用者でも容易に対象を選択し、データを登録できるよう設計した。

Figure 4.5: Katakana image viewing screen (initial state).

4.3.4 片仮名の閲覧機能

本システムでは、訓点資料における片仮名の出現箇所を一覧表示する機能を提供する。ユーザは特定の片仮名を選択し、その片仮名が資料内で使用されている箇所とその画像を確認できる。本機能は、片仮名の使用傾向や字体の違いを比較する際に役立つ。

閲覧画面の構成

閲覧画面は、片仮名の選択メニューと、選択された片仮名の出現箇所を一覧表示するエリアで構成される。初期状態では、Figure 4.5 に示すように、片仮名の選択メニューのみが表示されている。

ユーザが特定の片仮名を選択すると、Figure 4.6 に示すように、該当する片仮名の出現箇所がリスト形式で表示される。各エントリーには以下の情報が含まれる。

- **Page**: 片仮名が出現する資料の巻とページ番号（例: 巻 1：1 オ）
- **PageText**: 漢字の本文

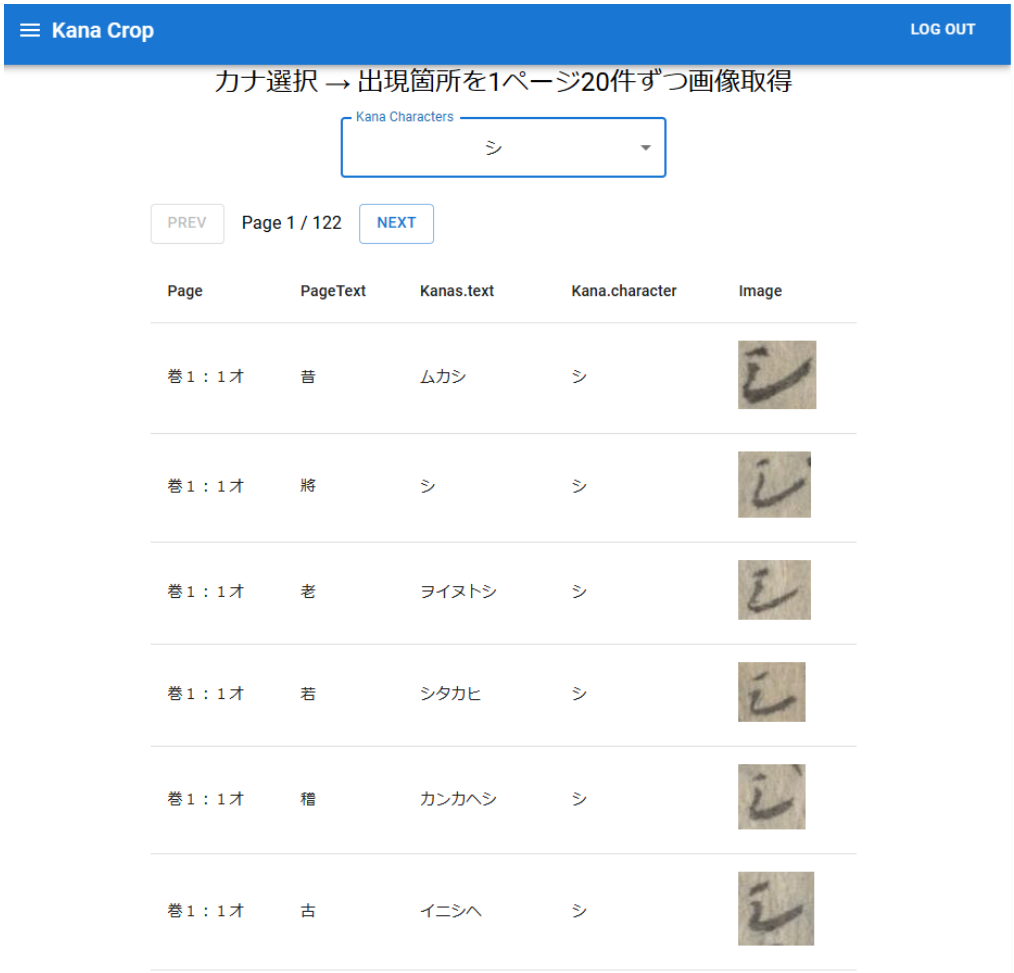

Figure 4.6: Katakana image viewing screen (viewing state).

- **Kanas.text**: 振り仮名の片仮名表記
- **Kana.character**: 特定の片仮名
- **Image**: 片仮名の切り出し画像

ページネーションの実装

片仮名の出現箇所が多いため、1 ページあたり 20 件ずつ表示するページネーションを導入している。ユーザは「NEXT」ボタンを押すことで次のページに進み、「PREV」ボタンで前のページに戻ることができる。この仕組みにより、データ量が多くても快適に閲覧できる。

本機能の実装により、片仮名の使用状況を視覚的に把握し、資料内の傾向分析を容易にすることが可能となった。

4.3.5　片仮名の登録

上記のように実装したアプリケーションを用いて、「尚書（古活字版第三種本）」の巻 1 の全ページの片仮名を切り出す作業を行った。片仮名の登録時には、以下のような観点でメモを付与し、データの正確性を担保した。

- **繰り返しの文字の処理**

  例えば、「ウヤウヤ」という表記がある場合、実際の資料では「ウヤ＋繰り返し文字」のように繰り返し文字が使用されることがある。そのため、このような表記の違いについてメモを付与し、後の解析で考慮できるようにした。

- **文字の汚れや異なる解釈の可能性**

  資料には、著しく汚れた部分や、文字の登録が異なると考えられるケースが存在した。そのため、こうした不確実な情報にはメモを付与し、後の分析時に参照できるようにした。

本研究では、以下のデータを登録した。

- 片仮名総数: **3,539 字**
- メモが付与された語群: **131 語**

この登録作業を通じて、資料内の片仮名データを統一的なフォーマットで管理し、今後の研究や応用に向けた基盤を構築した。

第5章　実験・評価

5.1　目的・実験内容

本実験では、作成した片仮名データベースの有効性を評価することを目的とし、特徴量の類似度を用いた誤登録文字の検出を行う。また、これによって歴史的資料において一般的に見られる字体の癖や手書きによる形状変化に対して、画像処理が有効に機能するかを検証する。この実験は、データベースに新たな文字を登録する状況に等しく、画像処理によって誤った文字の登録を防ぐことが可能かの検証も兼ねている。

5.1.1　実験の概要

本実験では、以下の手順で画像処理および評価を行う。

1. **画像の前処理**
    - 画像の二値化（大津の方法）
    - 画像のリサイズ（統一フォーマットへの変換）
2. **特徴量の抽出**
    - EasyOCR を用いた特徴量の取得
3. **類似文字検出の評価**
    - コサイン類似度を用いた異なる文字の判定

本実験では、**片仮名「シ」** を基準文字として選定し、類似文字の識別精度を評価する。片仮名「シ」は、登録作業中に「ミ」「ヲ」「レ」などの文字と誤登録されることが多く、検証対象として適していると判断した。

5.1.2　類似文字の混入と検出

1. **異なる文字を混入させた場合の識別精度評価**
    - **実験内容**: 正解データ群として 25 枚の明瞭な片仮名「シ」の画像を使用し、各試行で 1 枚の誤登録画像（「ミ」「ヲ」「レ」「ア」のいずれか）を混入させた上で、判定を行う。この試行を各文字ごとに異なる画像で 25 回ずつ（合計 100 回）実施し、識別精度を評価する。

- **目的**: 形状の類似性が異なる片仮名の識別精度を評価し、本データベースを用いた画像処理の有効性を検証する。

2. **画像のノイズ耐性の評価**

   - **実験内容**: 正解データ群を、文字のみが明瞭に映っている画像（クリーンデータ 25 枚）から、ノイズが含まれたものを含む無作為な片仮名「シ」の画像 25 枚に変更し、同様の判定を実施する。
   - **目的**: 画像の品質が識別精度に与える影響を評価し、ノイズ除去の必要性を検討する。

本実験の結果を通じて、画像処理の作成したデータベースへの適用可能性を検証し、今後の片仮名研究における有用性を示す。

5.2 事前準備

5.2.1 データ収集

本研究では、登録済みの片仮名画像データをデータベースからダウンロードし、実験用データとして利用する。使用する片仮名画像は、データベースの閲覧機能を活用し、画像の品質や適切なサンプル数を考慮した上で選定を行った。

実験において基準データとして用いる**片仮名「シ」**は、以下の 2 種類のデータセットを用意した。

- 無作為に選定した 25 枚の「シ」
- 文字のみが明瞭に映っている 25 枚の「シ」

また、比較対象となる**他の片仮名（「ミ」「ヲ」「レ」「ア」）**については、著しい汚れなどの影響を受けていない画像の中から無作為に選定し、データセットを構築した。

Figure 5.1 に示すように、(a) は文字のみが明瞭に映っている片仮名「シ」の画像、(b) は他の文字や汚れが含まれた片仮名「シ」の画像である。本実験では、これらを比較し、画像のノイズが識別精度に与える影響を評価する。

次節以降では、取得したデータに対する前処理について詳述する。

(a) clear "shi" (b) noisy "shi"

Figure 5.1: Comparison between a clear data and a noisy data.

（a）original　　（b）binary

Figure 5.2: Comparison between an original image and a binary image.

5.2.2　画像の二値化

片仮名画像の視認性を向上させ、特徴量抽出の精度を高めるために、以下の手順で二値化処理を行った。二値化により、画像内の文字と背景を明確に分離し、ノイズの影響を最小限に抑えることができる。

1. **画像の読み込みとグレースケール変換**

   入力画像はフルカラー（RGB）形式で格納されているため、まずグレースケールに変換し、輝度情報のみを保持する。

2. **ノイズ除去（ガウシアンフィルタ）**

   画像内の小さなノイズを低減し、二値化処理の精度を向上させるため、ガウシアンフィルタを適用する。

3. **大津の方法による二値化**

   大津の方法（Otsu's Method）を用いて、画像のヒストグラムから最適な閾値を自動的に決定し、二値化を行う。

4. **二値化画像の保存**

   処理後の画像は、オリジナルのファイル名に「_bin」を付加した形で保存する。

Figure 5.2 に、二値化処理前の画像と、二値化後の画像を示す。

5.2.3　画像の整形

次項で実施する特徴量抽出において、画像のサイズが統一されていない場合、特徴量の形式が揃わず、類似度の比較が正確に行えなくなる。そのため、すべての画像のサイズを統一し、特徴量の整合性を確保するため、以下の手順で事前にリサイズを行った。

1. **画像の読み込み**

   入力画像を RGB 形式で読み込む。

2. **アスペクト比を保持したリサイズ**

   画像の形状が歪まないように、アスペクト比を維持しながらリサイズを実施。

3. **余白の追加（パディング処理）**

   指定サイズ（128 × 128 px）に統一するため、不足分の余白を白色で補完。

4. **リサイズ後の画像の保存**

   処理後の画像を適切なフォルダに保存。

5.2.4 画像の特徴量抽出

片仮名画像の特徴を数値化するために、**EasyOCR** を用いて以下の手順で特徴量を抽出し、片仮名画像の形状情報を数値ベクトル化した。一般的な OCR の目的とは異なり、文字認識結果そのものを使用するのではなく、中間層の特徴マップを取得し、類似文字の識別に応用した。

1. **EasyOCR の初期化**

   日本語（ja）を対象として EasyOCR の Reader を設定し、特徴量の抽出を行う準備を実施。

2. **画像の読み込みと前処理**

   入力画像を RGB 形式で読み込み、ピクセル値を [0,1] の範囲に正規化した上で、テンソル形式に変換。

3. **特徴量の抽出**

   EasyOCR のベースネットワーク (basenet) の中間層（slice5）にフックを設定し、特徴マップを取得。

4. **特徴量の保存**

   抽出した特徴量をフラット化し、NumPy (.npy) 形式でファイルに保存。

5.2.5 異なる文字の検出方法

本研究では、片仮名画像の特徴量を比較し、異なる文字の混入を検出するために**コサイン類似度**を用いた手法を採用した。コサイン類似度は、2 つの特徴ベクトルの方向の類似度を測る指標であり、文字の形状が類似している場合には高い値を示し、異なる場合には低い値を示す。

検出の概要

異なる文字の検出手法は、以下の手順で実施した。

1. **特徴量の読み込み**
    - ベースデータを取得。
    - 誤登録の可能性があるデータを取得。
2. **コサイン類似度の計算**
    - ベースデータと誤登録データのほかのベクトルとのコサイン類似度を計算。
    - 各ベクトルの他のベクトルとの平均類似度を算出。
3. **異常データの検出**
    - 平均類似度が最も低いデータを異なる文字として特定。

これにより、文字群の中で最も他のベクトルとの類似度が低い文字を特定する。

5.3 明瞭な画像での実験

本実験では、明瞭な「シ」の特徴量（clear_shi）を基準として、「ア、ミ、ヲ、レ」の特徴量セット（a, mi, wo, re）から 1 枚を追加し、コサイン類似度を用いた異なる文字の検出を行った。各特徴量セットに対して実験を行い、検出成功率、平均類似度、及び検出の傾向を分析した。

5.3.1 各特徴量セットの実験結果

clear_shi + a

Table 5.1, 5.2 に明瞭なシにアを混入させた場合の結果を示す。結果より、シに混入させたすべてのアを正確に異なる文字として検出できていることが分かる。

clear_shi + mi

Table 5.3, 5.4 に明瞭なシにミを混入させた場合の結果を示す。結果より、シに混入させたすべてのミを正確に異なる文字として検出できていることが分かる。また、平均類似度が 0.7998 で範囲も 0.7126 〜 0.8493 と安定して高くミはアよりもシに近い文字であことが分かる。

Table 5.1: Results of clear_shi + a.

Error Index	**Result**	**Detected Index**	**Average Similarity**
a_1	○	a_1	0.8203
a_2	○	a_2	0.6831
a_3	○	a_3	0.8395
a_4	○	a_4	0.6203
a_5	○	a_5	0.8409
a_6	○	a_6	0.8179
a_7	○	a_7	0.8045
a_8	○	a_8	0.8453
a_9	○	a_9	0.7394
a_10	○	a_10	0.8201
a_11	○	a_11	0.8471
a_12	○	a_12	0.7952
a_13	○	a_13	0.5586
a_14	○	a_14	0.8230
a_15	○	a_15	0.8259
a_16	○	a_16	0.7833
a_17	○	a_17	0.8200
a_18	○	a_18	0.6958
a_19	○	a_19	0.6823
a_20	○	a_20	0.8378
a_21	○	a_21	0.8149
a_22	○	a_22	0.7525
a_23	○	a_23	0.8053
a_24	○	a_24	0.7917
a_25	○	a_25	0.8040

Table 5.2: Summary of clear_shi + a.

Item	**Result**
Total Trials	25
Successful Detections	25
Success Rate	100%
Mean of Similarity Scores	0.7787
Maximum Mean Similarity of Successes	0.8471
Minimum Mean Similarity of Failures	-
Similarity Range	0.5586 ∼ 0.8471

Table 5.3: Results of clear_shi + mi.

Error Index	**Result**	**Detected Index**	**Average Similarity**
mi_1	○	mi_1	0.8293
mi_2	○	mi_2	0.7820
mi_3	○	mi_3	0.8262
mi_4	○	mi_4	0.8066
mi_5	○	mi_5	0.7514
mi_6	○	mi_6	0.8038
mi_7	○	mi_7	0.8271
mi_8	○	mi_8	0.8493
mi_9	○	mi_9	0.8225
mi_10	○	mi_10	0.7126
mi_11	○	mi_11	0.7820
mi_12	○	mi_12	0.7277
mi_13	○	mi_13	0.8135
mi_14	○	mi_14	0.8393
mi_15	○	mi_15	0.7763
mi_16	○	mi_16	0.8065
mi_17	○	mi_17	0.7580
mi_18	○	mi_18	0.7636
mi_19	○	mi_19	0.8414
mi_20	○	mi_20	0.7977
mi_21	○	mi_21	0.8338
mi_22	○	mi_22	0.7999
mi_23	○	mi_23	0.8259
mi_24	○	mi_24	0.8139
mi_25	○	mi_25	0.8053

Table 5.4: Summary of clear_shi + mi.

Item	**Result**
Total Trials	25
Successful Detections	25
Success Rate	100%
Mean of Similarity Scores	0.7998
Maximum Mean Similarity of Successes	0.8493
Minimum Mean Similarity of Failures	-
Similarity Range	0.7126 ∼ 0.8493

Table 5.5: Results of clear_shi + wo.

Error Index	**Result**	**Detected Index**	**平均類似度**
wo_1	○	wo_1	0.6580
wo_2	○	wo_2	0.6409
wo_3	○	wo_3	0.8003
wo_4	○	wo_4	0.6084
wo_5	○	wo_5	0.6302
wo_6	○	wo_6	0.7126
wo_7	○	wo_7	0.7639
wo_8	○	wo_8	0.7950
wo_9	○	wo_9	0.7196
wo_10	○	wo_10	0.7439
wo_11	○	wo_11	0.8127
wo_12	○	wo_12	0.6988
wo_13	○	wo_13	0.7097
wo_14	○	wo_14	0.8373
wo_15	○	wo_15	0.7658
wo_16	○	wo_16	0.6432
wo_17	○	wo_17	0.6887
wo_18	○	wo_18	0.8350
wo_19	○	wo_19	0.6638
wo_20	○	wo_20	0.7769
wo_21	○	wo_21	0.7258
wo_22	○	wo_22	0.7700
wo_23	○	wo_23	0.7493
wo_24	○	wo_24	0.7572
wo_25	○	wo_25	0.5552

Table 5.6: Summary of clear_shi + wo.

Item	**Result**
Total Trials	25
Successful Detections	25
Success Rate	100%
Mean of Similarity Scores	0.7225
Maximum Mean Similarity of Successes	0.7225
Minimum Mean Similarity of Failures	-
Similarity Range	0.5552 ～ 0.8373

Table 5.7: Results of clear_shi + re.

Error Index	Result	Detected Index	平均類似度
re_1	×	clear_shi_10	0.8635
re_2	○	re_2	0.7866
re_3	○	re_3	0.8248
re_4	×	clear_shi_10	0.8647
re_5	×	clear_shi_10	0.8649
re_6	○	re_6	0.8547
re_7	○	re_7	0.8590
re_8	×	clear_shi_10	0.8640
re_9	×	clear_shi_10	0.8650
re_10	×	clear_shi_10	0.8646
re_11	○	re_11	0.8424
re_12	×	clear_shi_10	0.8649
re_13	×	clear_shi_10	0.8656
re_14	×	clear_shi_10	0.8639
re_15	×	clear_shi_10	0.8639
re_16	○	re_16	0.8240
re_17	×	clear_shi_10	0.8642
re_18	×	clear_shi_10	0.8666
re_19	×	clear_shi_10	0.8681
re_20	×	clear_shi_10	0.8663
re_21	×	clear_shi_10	0.8658
re_22	×	clear_shi_10	0.8632
re_23	×	clear_shi_10	0.8674
re_24	○	re_24	0.8626
re_25	×	clear_shi_10	0.8660

clear_shi + wo

Table 5.5, 5.6 に明瞭なシにヲを混入させた場合の結果を示す。結果より、シに混入させたすべてのヲを正確に異なる文字として検出できていることが分かる。しかし、平均類似度が 0.7225 で範囲も 0.5552 〜 0.8373 とばらつきがあり、ヲはアよりもシと判別がつきやすい文字であることが分かる。

clear_shi + re

Table 5.7, 5.8 に明瞭なシにレを混入させた場合の結果を示す。結果より、全体の 72%のレが正確に異なる文字として検出できなかったことが分かる。また、平均類似度が 0.8571 と非常に高いことから、レが非常にシと判別がつきにくい文字であることが分かる。

Table 5.8: Summary of clear_shi + re.

Item	Result
Total Trials	25
Successful Detections	7
Success Rate	28%
Mean of Similarity Scores	0.8571
Maximum Mean Similarity of Successes	0.8626
Minimum Mean Similarity of Failures	0.8632
Similarity Range	0.7866 〜 0.8681

5.4 無作為な画像での実験

本実験では、ノイズを含む無作為に選んだ「シ」の特徴量（noisy_shi）を基準として、「ア、ミ、ヲ、レ」の特徴量セット（a, mi, wo, re）を追加し、コサイン類似度を用いた異なる文字の検出の有効性を評価した。各特徴量セットに対して実験を行い、成功率、平均類似度、および検出の傾向を分析する。

5.4.1 各特徴量セットの実験結果

noisy_shi + a

Table 5.9, 5.10 にノイズ含むシにアを混入させた場合の結果を示す。結果より、シに混入させたアのうち 72%が異なる文字として検出できていないことが分かる。平均類似度も 0.7381 とあまり高くなく、ベースデータのノイズが大きな原因であると考えられる。

noisy_shi + mi

Table 5.3, 5.12 に明瞭なシにミを混入させた場合の結果を示す。結果より、シに混入させたミの 80%が異なる文字として検出できていないことが分かる。また、今回の平均類似度も 0.7579 と高くない。また、アと比較して誤検知が多いことから、ノイズの入ったシに対してもミはアよりも近い文字であると考えられる。

noisy_shi + wo

Table 5.13, 5.14 にノイズを含むシにヲを混入させた場合の結果を示す。結果より、シに混入させたうち 54%のヲが正確に異なる文字として検出されており、後述のレを含めた中で、最もノイズを含むシと判別がしやすい文字であることが分かる。

Table 5.9: Results of noisy_shi + a.

Error Index	**Result**	**Detected Index**	**Mean Similarity**
a_1	×	noisy_shi_6	0.7628
a_2	○	a_2	0.6779
a_3	×	noisy_shi_6	0.7622
a_4	○	a_4	0.6383
a_5	×	noisy_shi_6	0.7629
a_6	×	noisy_shi_6	0.7606
a_7	×	noisy_shi_6	0.7629
a_8	×	noisy_shi_6	0.763
a_9	○	a_9	0.7424
a_10	×	noisy_shi_6	0.7621
a_11	×	noisy_shi_6	0.7627
a_12	×	noisy_shi_6	0.7639
a_13	○	a_13	0.5624
a_14	×	noisy_shi_6	0.7624
a_15	×	noisy_shi_6	0.7624
a_16	×	noisy_shi_6	0.7598
a_17	×	noisy_shi_6	0.7622
a_18	○	a_18	0.6927
a_19	○	a_19	0.6778
a_20	×	noisy_shi_6	0.7641
a_21	×	noisy_shi_6	0.7610
a_22	○	a_22	0.7398
a_23	×	noisy_shi_6	0.7610
a_24	×	noisy_shi_6	0.7613
a_25	×	noisy_shi_6	0.7626

Table 5.10: Summary of noisy_shi + a.

Item	**Result**
Total Trials	25
Successful Detections	7
Success Rate	28%
Mean of Similarity Scores	0.738
Maximum Mean Similarity of Successes	0.7424
Minimum Mean Similarity of Failures	0.7598
Similarity Range	0.5624 ∼ 0.7641

Table 5.11: Results of noisy_shi + mi.

Error Index	**Result**	**Detected Index**	**平均類似度**
mi_1	○	mi_1	0.7157
mi_2	×	noisy_shi_6	0.7597
mi_3	○	mi_3	0.7253
mi_4	×	noisy_shi_6	0.7630
mi_5	×	noisy_shi_6	0.7632
mi_6	×	noisy_shi_6	0.7599
mi_7	×	noisy_shi_6	0.7619
mi_8	○	mi_8	0.7556
mi_9	○	mi_9	0.7553
mi_10	×	noisy_shi_6	0.7628
mi_11	×	noisy_shi_6	0.7625
mi_12	×	noisy_shi_6	0.7609
mi_13	×	noisy_shi_6	0.7623
mi_14	×	noisy_shi_6	0.7626
mi_15	×	noisy_shi_6	0.7617
mi_16	×	noisy_shi_6	0.7615
mi_17	×	noisy_shi_6	0.7624
mi_18	×	noisy_shi_6	0.7606
mi_19	×	noisy_shi_6	0.7620
mi_20	×	noisy_shi_6	0.7625
mi_21	○	mi_21	0.7569
mi_22	×	noisy_shi_6	0.7622
mi_23	×	noisy_shi_6	0.7627
mi_24	×	noisy_shi_6	0.7625
mi_25	×	noisy_shi_6	0.7615

Table 5.12: Summary of noisy_shi + mi.

Item	**Result**
Total Trials	25
Successful Detections	5
Success Rate	20%
Mean of Similarity Scores	0.7579
Maximum Mean Similarity of Successes	0.7569
Minimum Mean Similarity of Failures	0.7599
Similarity Range	0.7157 ～ 0.7632

Table 5.13: Results of noisy_shi + wo.

Error Index	Result	Detected File	Average Similarity
wo_1	×	noisy_shi_6	0.7636
wo_2	×	noisy_shi_6	0.7635
wo_3	○	wo_3	0.7097
wo_4	×	noisy_shi_6	0.7613
wo_5	○	wo_5	0.7522
wo_6	×	noisy_shi_6	0.7613
wo_7	×	noisy_shi_6	0.7647
wo_8	×	noisy_shi_6	0.7645
wo_9	○	wo_9	0.6067
wo_10	○	wo_10	0.6811
wo_11	○	wo_11	0.6728
wo_12	×	noisy_shi_6	0.7630
wo_13	○	wo_13	0.6355
wo_14	○	wo_14	0.6652
wo_15	○	wo_15	0.7329
wo_16	×	noisy_shi_6	0.7615
wo_17	×	noisy_shi_6	0.7619
wo_18	○	wo_18	0.7517
wo_19	×	noisy_shi_6	0.7608
wo_20	○	wo_20	0.7199
wo_21	○	wo_21	0.7370
wo_22	×	noisy_shi_6	0.7633
wo_23	×	noisy_shi_6	0.7623
wo_24	○	wo_24	0.6733
wo_25	○	wo_25	0.7368

Table 5.14: Summary of noisy_shi + wo.

Item	Result
Total Trials	25
Successful Detections	13
Success Rate	54%
Mean of Similarity Scores	0.7291
Maximum Mean Similarity of Successes	0.7613
Minimum Mean Similarity of Failures	0.7608
Similarity Range	0.6067 ∼ 0.7647

Table 5.15: Results of noisy_shi + re.

Error Index	Result	Detected Index	Average Similarity
re_1	×	noisy_shi_6	0.7593
re_2	○	re_2	0.7265
re_3	×	noisy_shi_6	0.7653
re_4	×	noisy_shi_6	0.7580
re_5	×	noisy_shi_6	0.7584
re_6	×	noisy_shi_6	0.7631
re_7	×	noisy_shi_6	0.7627
re_8	×	noisy_shi_6	0.7589
re_9	×	noisy_shi_6	0.7592
re_10	×	noisy_shi_6	0.7581
re_11	×	noisy_shi_6	0.7630
re_12	×	noisy_shi_6	0.7597
re_13	×	noisy_shi_6	0.7598
re_14	×	noisy_shi_6	0.7602
re_15	×	noisy_shi_6	0.7602
re_16	×	noisy_shi_6	0.7631
re_17	×	noisy_shi_6	0.7607
re_18	×	noisy_shi_6	0.7573
re_19	×	noisy_shi_6	0.7595
re_20	×	noisy_shi_6	0.7584
re_21	×	noisy_shi_6	0.7627
re_22	×	noisy_shi_6	0.7612
re_23	×	noisy_shi_6	0.7585
re_24	×	noisy_shi_6	0.7587
re_25	×	noisy_shi_6	0.7587

Table 5.16: Summary of noisy_shi + re.

Item	Result
Total Trials	25
Successful Detections	1
Success Rate	4%
Mean of Similarity Scores	0.7588
Maximum Mean Similarity of Successes	0.7265
Minimum Mean Similarity of Failures	0.7631
Similarity Range	0.7265 ∼ 0.7653

noisy_shi + re

Table 5.15, 5.16 にノイズを含むシにレを混入させた場合の結果を示す。結果より、96%のレを正確に異なる文字として検出できなかったことが分かる。類似度のばらつきもなく、ノイズを含んでもレがシと非常に判別がつきにくい文字であることが分かる。

5.5 評価・考察

明瞭なシに対する実験では a, mi, wo セットでは、すべての文字が異なる文字として検出され、re セットでは、多くの文字が異なる文字としての検出されなかった。また、成功時の最大類似度や失敗時の最小類似度を見ると、明瞭なシの画像群は 0.86 以上の類似度を持っており、EasyOCR を用いた画像認識はうまく機能していると考えられる。

一方、平均類似度に着目すると、 レ ＞ ミ ＞ ア ＞ ヲ の順でシに近い文字と判定されていることが分かる。アとミは形状から考えて妥当な類似度であるが、レとヲに関しては直観に反する結果となった。このことからは、点などの細かい形状の情報の重みよりも、「はらい」などによって読み取られる筆順や情報の重みが大きくなっていると考えられる。

実際に Figure 5.3 を見ると、点の数などの形状では、レよりもミ、ミよりもヲの方がシに近く見える。しかしその一方でシの下から上向きの「はらい」に相当する部分に着目すると、レはかなりシに近い動きをしており、ミも左側から右向きという筆運びの向きは一致しているのに対し、アとヲは上から下向きに「はらい」が行われているうえ、ヲに関しては上の点の筆運びもシと逆向きになっていることが分かる。

ノイズを含むシに対する実験では、すべてのセットで異なる文字を検出することが困難であった。類似度の平均や成功時の最大値や失敗値の最小値を見ても、シ全体の類似度があまり高くなっておらず、EasyOCR を用いた画像認識がうまく機能していないことが分かる。このことから、本研究で開発したデータベースを画像処理に活用するにはノイズの除去が課題であることが分かった。

しかし、ヲやレの検知率は他と比較して明確に高いまたは低い傾向が表れており、ノイズが含まれたとしても完全に文字の判別ができなくなったわけではないと考えられる。

これらの課題に対するアプローチとして、類似度判定の変更や、複数の判定基準を組み合わせることで精度の向上を図ることが考えられる。しかし、ノイズによる影響に関しては、実験全体を通してベースデータの類似度低下が原因であることが推測されるため、このアプローチでは効果が期待できない。

ノイズに対するアプローチとして、ノイズを除去することが考えられるが、本実験におけるノイズとは他の文字の映り込みなどであるため色や太さでの除去が難しく、画像切り出しの範囲選択を柔軟にする等のアプローチが必要になると考えられる。

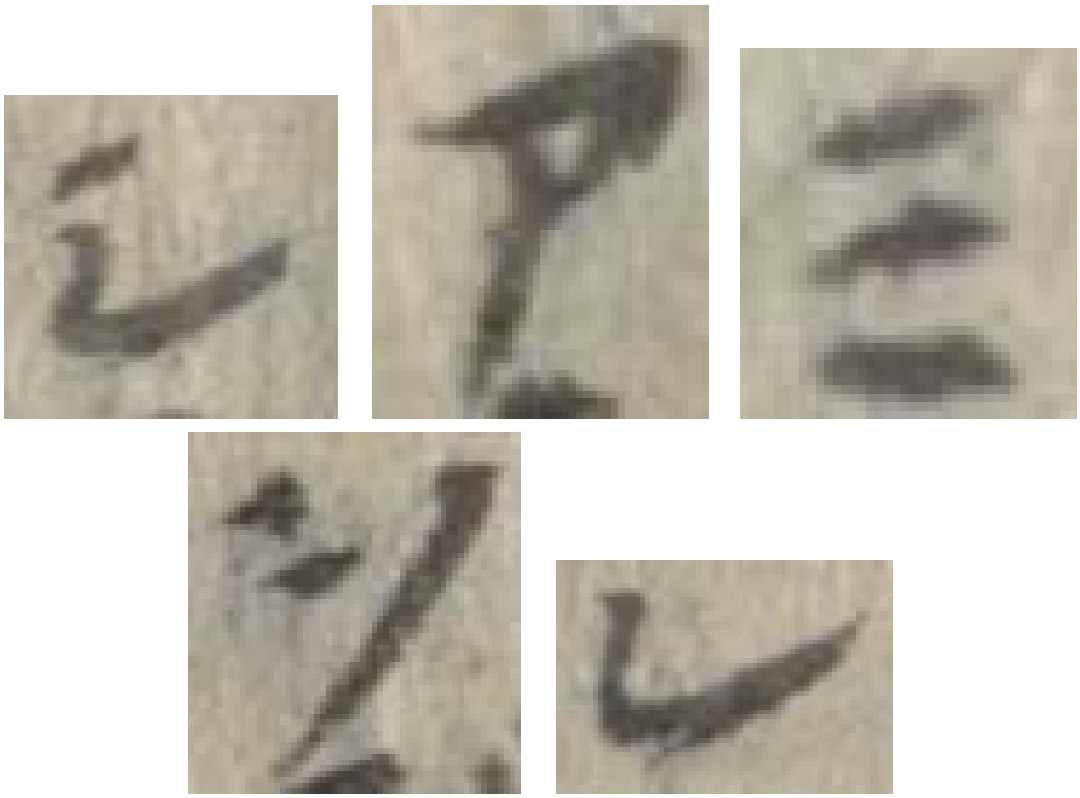

Figure 5.3: Image of shi, a, mi, wo, re.

第6章　結論

6.1　結論

本研究では PHP による API サーバと、React(TypeScript)+Next.js+Material UI を用いたアクセシビリティの高い Web アプリケーションを開発し、「尚書（古活字版 第三種本）」の訓点資料データベースを拡張することで、一次資料に基づいた片仮名に関するデータベースシステムを開発した。これによって片仮名の情報や画像データ、それに対する補足情報などをデータベースに保存できるようになり、Web アプリケーションを用いて片仮名の画像や補足の登録、それらの閲覧が可能になった。

また、データベースに登録した画像を用いた実験を行い、大津の方法で二値化した画像をから EasyOCR を用いた特徴量抽出を行い、コサイン類似度を用いて処理を行うことで、ノイズがあまり含まれない画像であれば画像の検知などへの応用が見込めることを示した。一方で、一部類似した文字やノイズによる影響が大きく、本データベースを画像処理に応用するためには、これらの問題を解決することが今後の課題となった。

謝辞

最後に、本研究を進めるにあたり、ご多忙中にもかかわらず多大なご指導をしていただきました指導教員の田島孝治先生、主査として論文に細やかなご指摘をいただいた出口利憲先生、また、ともに勉学に励んだ研究室のメンバーに厚く御礼を申し上げます。

参考文献

[1] 春日政治, “西大寺本金光明最勝王経古点の国語学的研究”, 斯道文庫, 1942.

[2] 築島裕, “興福寺本大慈恩寺三蔵法師伝古点の国語学的研究（訳文篇・索引篇・研究篇）”, 東京大学出版会, 1965-1967.

[3] 小林芳規（太田次男と共著）, “神田本白氏文集の研究”’, 勉誠社, 1982.

[4] 林昌哉, 田島孝治, 堤智昭, 高田智和, 小助川貞次, “訓点資料の加点情報計量のためのデータ構造－国立国語研究所蔵「尚書（古活字版）」を対象として－”, じんもんこん 2017 論文集, Vol.2017, pp.45-52, 2017.

[5] 田島孝治, 堤智昭, 高田智和, “訓点資料の書き下し文自動生成を目的としたヲコト点を中心とする訓点の計量分析”, 情報処理学会論文誌, vol.61 No.2, pp.162-170, 2020.

[6] 田島 孝治, Baptiste Jannequin, 堤 智昭, 高田 智和, “IIIF Viewer と連携可能な訓点資料の加点情報データベースの試作”, じんもんこん 2019 論文集, pp.109-114, 2019.

[7] 苫米地 康太, “訓点資料画像の文字位置検出とデータベースへの追加”, 情報処理学会研究報告 (Web), 2023 巻 CH-132 号, pp.No.14 1-4, 2023.

[8] 国立国語研究所, “尚書（古活字版 第三種本）訓点情報データベース”（2024/1/25 閲覧）.

`https://cid.ninjal.ac.jp/kunten-syousyo3/`

[9] 国立国語研究所学術情報リポジトリ, “訓点資料の構造化記述 成果報告書”（2024/1/25 閲覧）.

`https://repository.ninjal.ac.jp/record/2657/files/crpr_12-08.pdf`

[10] TypeScript, “JavaScript With Syntax For Types.”（2024/1/25 閲覧）.

`https://www.typescriptlang.org`

[11] React, “React”（2024/1/25 閲覧）.

`https://ja.react.dev/`

[12] Next.js ドキュメント日本語翻訳プロジェクト, “Next.js 日本語ドキュメント”（2024/1/25 閲覧）.

`https://nextjsjp.org/`

[13] MUI, “The React component library you always wanted”（2024/1/25 閲覧）.
https://mui.com

[14] AWS, “LAMP スタックとは？”（2024/1/25 閲覧）.
https://aws.amazon.com/jp/what-is/lamp-stack/

[15] Canonical Ltd., “Ubuntu official documentation,”（2024/1/25 閲覧）.
https://help.ubuntu.com/

[16] Apache Software Foundation, “Apache HTTP Server Project,”（2024/1/25 閲覧）.
https://httpd.apache.org/

[17] MariaDB Foundation, “MariaDB - Open Source Database,”（2024/1/25 閲覧）.
https://mariadb.org/

[18] PHP 公式ドキュメント, “PHP ドキュメント”（2024/1/25 閲覧）.
https://www.php.net/manual/ja/index.php

[19] マクセルフロンティア株式会社, “画像の 2 値化 (大津の 2 値化)”（2024/1/25 閲覧）.
https://www.frontier.maxell.co.jp/blog/posts/4.html

[20] Jaided AI, “EasyOCR Documentation”（2024/1/25 閲覧）.
https://www.jaided.ai/easyocr/documentation/

[21] 数学の景色, “コサイン類似度とは〜定義と具体例〜”（2024/1/25 閲覧）.
https://mathlandscape.com/cos-similar/