平成 30 年度　卒業論文

顔認証による Web アプリケーションログイン実現のためのフレームワーク作成

平成 31 年 2 月 21 日

15108059

白土　直樹

指導教員　張　力峰 准教授

九州工業大学工学部

電気電子工学科

電子工学コース

概要

現在，多種多様な Web アプリケーションが世の中に存在しており，多くの人が利用している．しかし，Web アプリケーションには複数の情報セキュリテイ問題が生じる恐れがある．その問題の中に，他者が正規ユーザになりすまして正規ユーザの Web サーバに侵入するセッション問題がある．また，Web アプリケーションユーザの高齢化が進んでおり，今後は現在よりもユーザのログイン情報の暗記，入力の負担も増加すると考えられる．

本研究では，正規ユーザがログインする際，Web カメラを用いてサーバー上で撮影した顔画像と自身の登録した顔画像とを比較しログインを行う仕組みを作成することにより，従来の認証システムよりもユーザのログイン情報の暗記，入力の負担を軽減する新たな仕組みを実現する．

目次

第 1 章

序論

1.1　研究背景

近年の情報化に伴い, 様々な Web アプリケーションが存在しており，多くの人がインターネットを介して多種多様なサービスを利用している．Web アプリケーションとは，Web ブラウザから利用可能なアプリケーション，サービスのことを指し，ユーザ側のブラウザとサーバ側のアプリケーションサーバなどのプログラムが互いに通信を行うことでサービスを実現している．しかし近年では Web アプリケーションそのものを狙った攻撃が急増しており，他者のなりすましログインなどのセキュリティに関する複数の問題が存在し，Web アプリケーションの脆弱性が顕著にみられる．セキュリティを考慮するため，ユーザ認証として ID とパスワード（ログイン情報）を用い，ユーザ毎のサービスを提供する Web アプリケーションが主流になっている．しかし，パスワード認証が増えるにつれて，ユーザのログイン情報の暗記，入力の負担も増加する．そこで私は既存の Web アプリケーションのフレームワークを利用し，新しいユーザの認証システムを顔画像を比較する手法を用いて作成することにした．

1.2　高齢化

現在，日本では高齢者の人口割合が増加しており，今後も高齢者の人口および，総人口に対する高齢者比率はますます増加する傾向にある．以下，図 1.1 に日本の高齢化の推移と将来推計を示す．

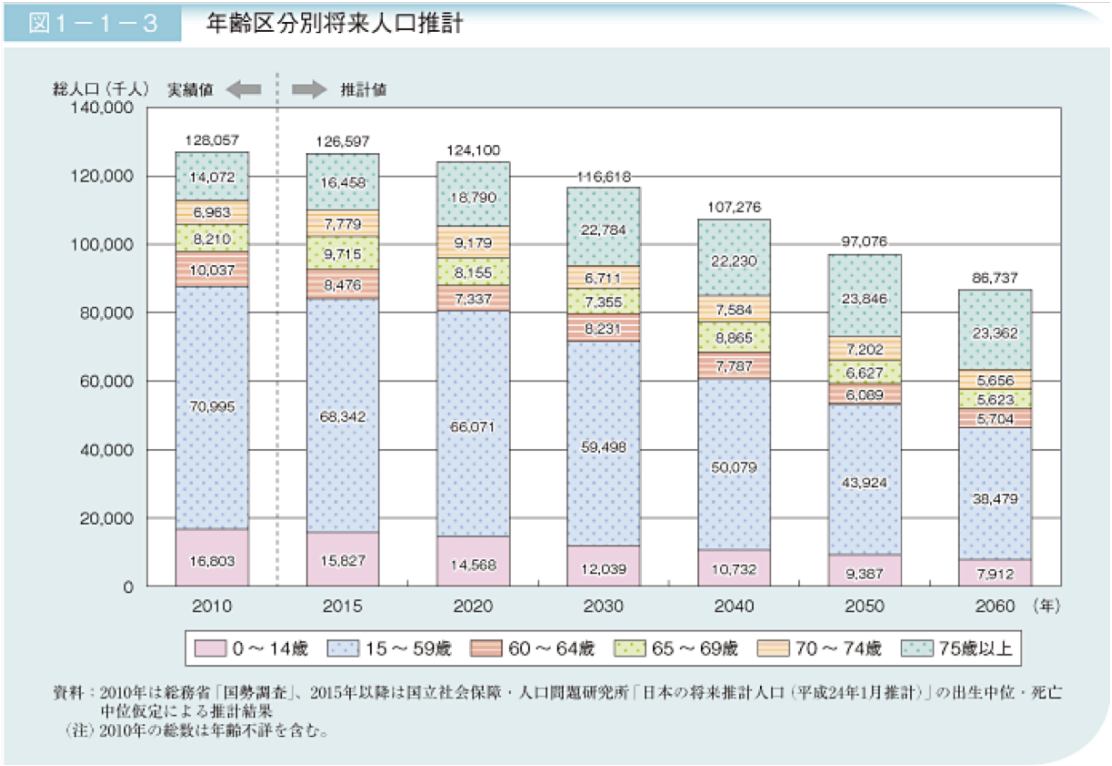

図 1.1: 日本の高齢化の推移と将来推計

出典 https://www8.cao.go.jp/index.html

高齢者人口そのものは団塊の世代との兼ね合わせもあり，2040 年過ぎ（2042 年）に 3,935 万人でピークを迎えると推測されているものの，それより下の世代，そして総人口も減少をしているため，高齢者比率は増加している．2035 年には高齢者比率 33.3 %と，ほぼ 3 人に 1 人が 65 歳以上の高齢者に，そして 2060 年には 36.7 %に達し，国民の約 2.7 人に 1 人が 65 歳以上の高齢者となることが予想される．

また現在の日本国内におけるインターネット普及率の推移状況についてみていく．図 1.2，図 1.3 は，それぞれ国内でのインターネット普及率の推移，直近 2 年間における世代別の普及率の推移を示す．2013 年から普及率８０％を突破しており，実に 5 人に 4 人以上が国内での生活の中でインターネットを介した Web アプリケーションの利用を求めていることがわかる．

図 1.3 では世代毎に様々な変化が見られる中，高年齢層と呼ばれる 50,60 代の方のインターネット利用率が著しく増加している．すなわち，Web アプリケーションのユーザのおよそ半分が高齢者となり，現在よりいっそうユーザのログイン情報の暗記，入力の負担が増加すると考えられる．

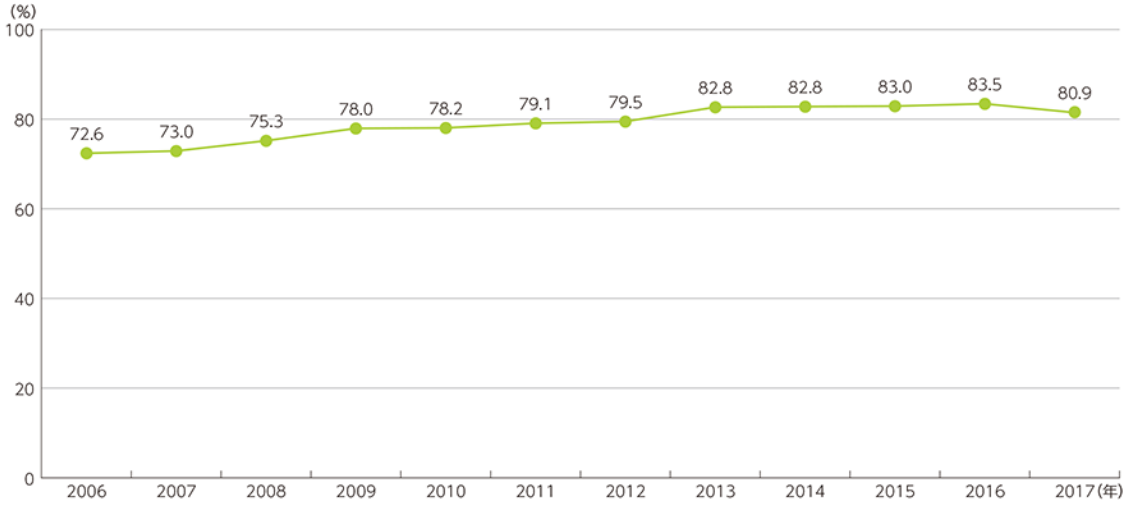

図 1.2: 国内利用者数および人口普及率の推移

出典 http://www.soumu.go.jp/johotsusintokei/statistics/statistics05.html

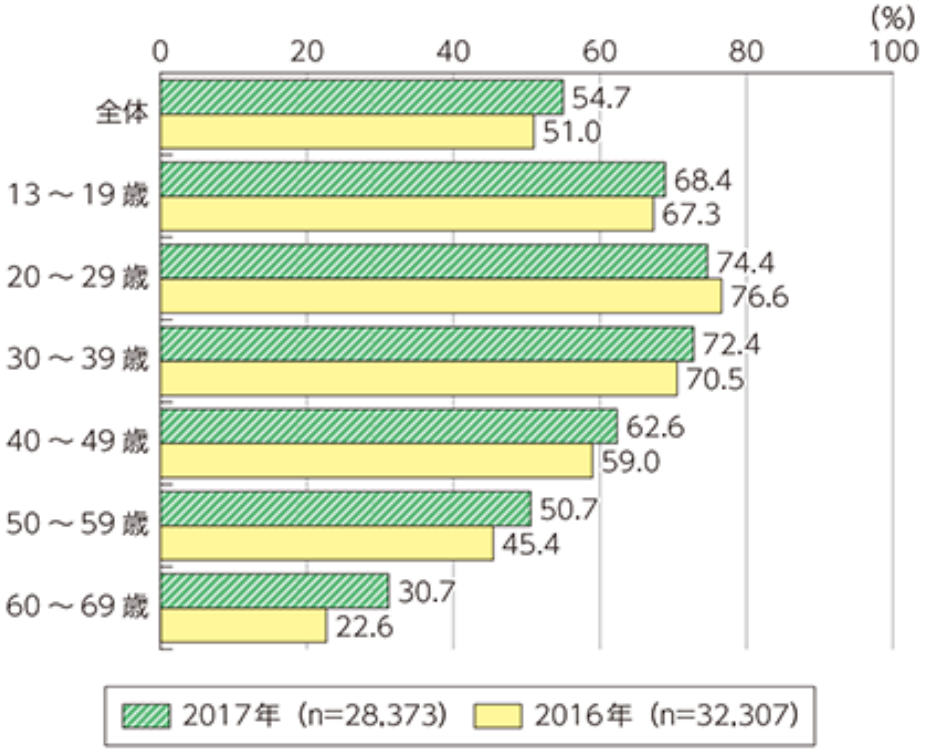

図 1.3: 国内世代別インターネット利用率の推移

出典 http://www.soumu.go.jp/johotsusintokei/statistics/statistics05.html

1.3 研究目的

本研究では，拡大が予測されるユーザの年齢層に対応しながらも，Web アプリケーションの脆弱性を解決すべく，既存の Web アプリケーションフレームワークを用いて，ユーザのログイン情報の暗記，入力の負担を軽減するシステムを顔画像を比較する手法を用いて，それぞれのシステムの有用性を確認するとともに，新しいユーザの認証システムを実現することを目的とする．

第 2 章

提案

2.1　画像処理を用いた認証システム

近年の情報化に伴い，本人認証の重要度はますます大きくなっている．コンピュータやスマートフォンへのログインや，銀行の ATM 利用時，さらには，帰宅時にさえ本人認証を行うことがあり，今後もその機会は多くなっていくと考えられる．本人認証にはいくつかの方法があり，所有物認証，知的認証，生体認証などがある．この中でも，本人の身体的特徴を利用する生体認証は本人以外の人間が成りすますことが難しくなり，今後の社会においても生体認証は高い需要性を保つと考えられる．

本研究で開発した顔画像に対する画像処理を用いた認証システムとは，サーバ上で撮影した顔画像から輝度値を算出し識別するシステムである．あらかじめユーザの顔画像を登録しておき，認証の際，入力された顔画像から得られる特徴点が登録された顔画像と一致しているかなどを判断し，ユーザの認証をすることができる．

2.2　Web アプリケーションの仕組み

Web アプリケーションの構成は大きく分けて，クライアントサイドとサーバサイドの 2 つに分けられる．Web アプリケーションの仕組みを図 2.1 に示す．

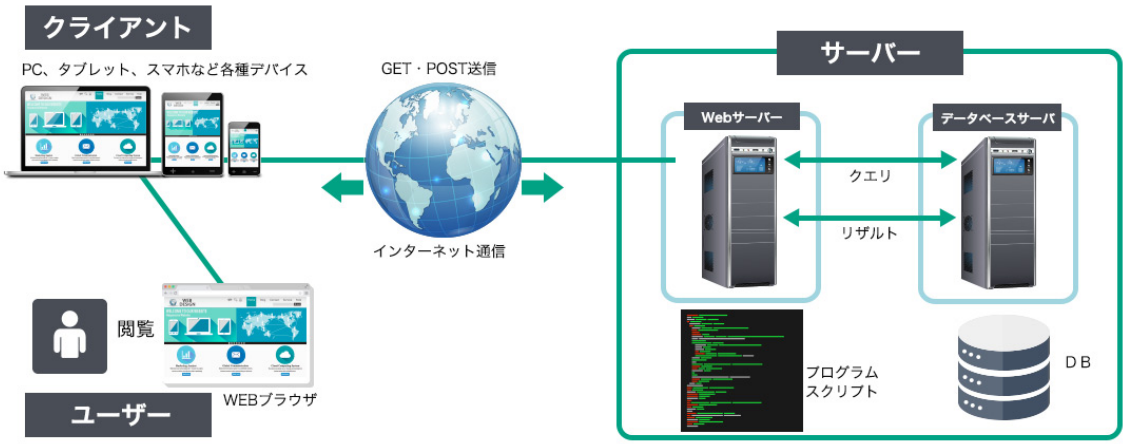

図 2.1: Web アプリケーションの構造

出典 http://www.claire.co.jp/service/webappli

2.2.1 クライアントサイド

クライアントサイドとは複数のコンピュータが接続されたシステムにおいて，クライアントサーバ関係におけるクライアントによって実行される操作，処理のことを指す．ここでいうクライアントとは，ユーザのローカルコンピュータの上で実行され必要に応じてサーバに接続する Web ブラウザのようなコンピュータアプリケーションのことであり，サーバからサービスを受ける側を指す．

クライアントサイドの特徴として，Web サーバーに保存されたプログラムがクライアントのメモリに読み込まれ，Web ブラウザがそれを実行し，処理結果も，クライアントのブラウザに表示される点が挙げられる．短所として，Web サーバーからクライアントへ送られるのは，処理結果ではなくプログラムそのものとなるため，プログラムの読み込みに時間がかかる．また，プログラムの実行速度はクライアントの処理能力に左右される．

2.2.2 サーバサイド

サーバサイドとは Web サーバ上で動作し，Web サーバ上でプログラムの実行が要求されるたびに，結果を Web ブラウザに対して送信するようなプログラムである．

特徴として，Web サーバ上でプログラムが実行され，その処理結果がクライアントに送られてブラウザに表示される．そのため，サーバサイドで開発を行う利点として，ブラウザを実行

しているクライアントの側には，ほとんど負荷がかからない点が挙げられる．

本研究ではデータベースの操作や画像ファイルの読み書きのしやすいサーバサイドで Web アプリケーションを開発する．

2.3 Web アプリケーションフレームワーク

Web アプリケーションフレームワークとは，動的な Web サイト，Web アプリケーション，Web サービスの開発をサポートするために設計されたアプリケーションフレームワークである． フレームワークの目的は，Web 開発で用いられる共通した作業に伴う労力を軽減することである．たとえば，多数のフレームワークがデータベースへのアクセスのためのライブラリや，テンプレートエンジンや，セッション管理を提供することで，コードの再利用を促進させるものもある．本研究では，作業の簡略化が可能な点と，開発を比較的に容易に行える点，さらに無料で誰でも使用することができる点から，既存の Web アプリケーションフレームワークを編集して新しいユーザの認証システムを作成する．

近年，開発を行う際に利用することができる Web 言語は増え続ける一方であり，Web アプリケーションフレームワークの種類によって使用されるプログラミング言語の種類も異なってくる．現在，存在する Web 系称する言語は 50 種類を超えると言われている．今回は，特に代表的な『Java』,『PHP』,『Ruby』の 3 種類の言語に着目する．以下に Web アプリケーションフレームワークとプログラミング言語の関係性について以下の表にまとめる．

	効率	性能	コスト	特徴
JSF	○	○	△	MVC モデル 2 による Web アプリケーション・「UI コンポーネント」で Web ページを構成
Spring FW	○	△	○	動作に必要な他のクラスを Spring で生成
Struts	○	○	○	HTML とコードを分けるカスタムタグ
Wicket	○	△	○	HTML ファイルをそのままテンプレートとして使用できる
Play FW	○	△	○	Scala と Java 言語で書かれたオープンソース

図 2.2: Java フレームワーク　出典 https://furien.jp/columns/46/

	効率	性能	コスト	特徴
CakePHP	○	○	○	PEAR などのライブラリがなくても動作する
Zend FW	○	△	○	従うべき開発パラダイムや開発パターンが無く、自由度が高い
Laravel	△	○	○	独特の特徴が無い・頻繁に機能追加あり
Codeigniter	○	△	○	Ruby on Rails、Mojavi の思想を引き継いでいる
symfony	○	○	△	軽量なフレームワークで、処理速度が速い

図 2.3: PHP フレームワーク　出典 https://furien.jp/columns/46/

	効率	性能	コスト	特徴
Ruby on Rails	○	○	○	少ないコードで簡単に開発できるよう考慮されている
Sinatra	△	○	○	小さく、柔軟性があるプログラミングが可能
Padrimo	○	○	○	Sinatra ベースに MVC 構造やヘルパー、国際化機能、テスト自動生成などの機能を追加

図 2.4: Ruby フレームワーク　出典 https://furien.jp/columns/46/

本研究では，サーバサイドで動的な Web ページ作成するための機能を多く備える PHP を使用する．PHP を使用するメリットとしては，以下の点が挙げられる．

- サーバサイド Web アプリケーション構築のための豊富な組み込み関数を持つ．
- オープンソースで誰でも使用可能である．
- データベースへのアクセスが容易である．
- 多くのオープンソースのフレームワークやライブラリが利用可能である．

またフレームワークは『Laravel』を使用する．Laravel を使用するメリットとしては，以下の点が挙げられる．

- コードがすぐに書き始められるうえに書きやすい．
- スムーズに開発することが可能である．
- 癖が無く勉強コストも少ないので習得しやすい．

2.4　Laravel について

Laravel（ララベル）は，Microsoft の .NET の開発に関わっていた Taylor Otwel l が開発した PHP のフレームワークである．Larave は，フリーのオープンソースであり，MIT ライセンスのため，自由に利用することができる．PHP のフレームワークは複数存在しているが，その中でも Laravel は世界的に利用されている．各 PHP フレームワークの検索数について以下の図 2.5 に示す．

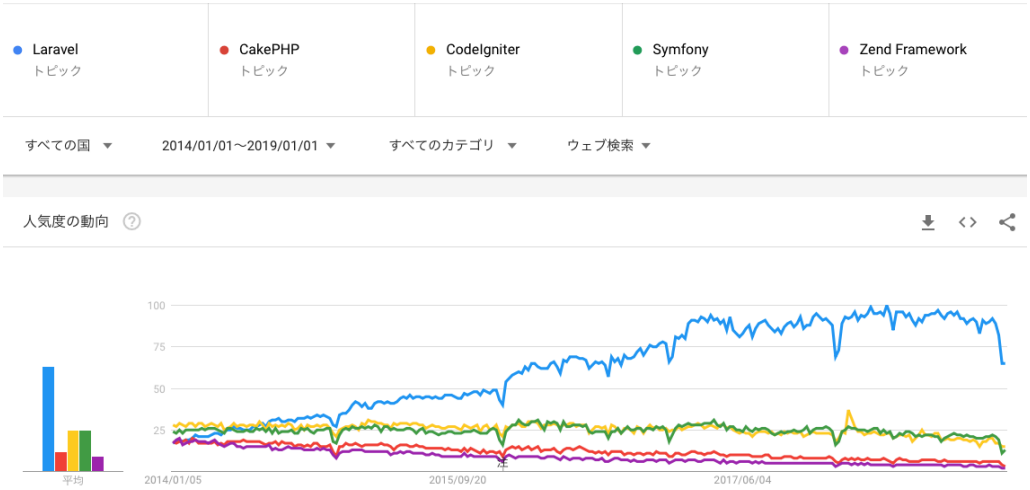

図 2.5: 世界のインターネット (google) 上での PHP フレームワークの検索数

出典 https://trends.google.co.jp

グラフに示したように PHP のフレームワークの中でも最も利用されていることがわかる．さらに日本国内における PHP フレームワークの動向についても以下の図 2.6 に示す．

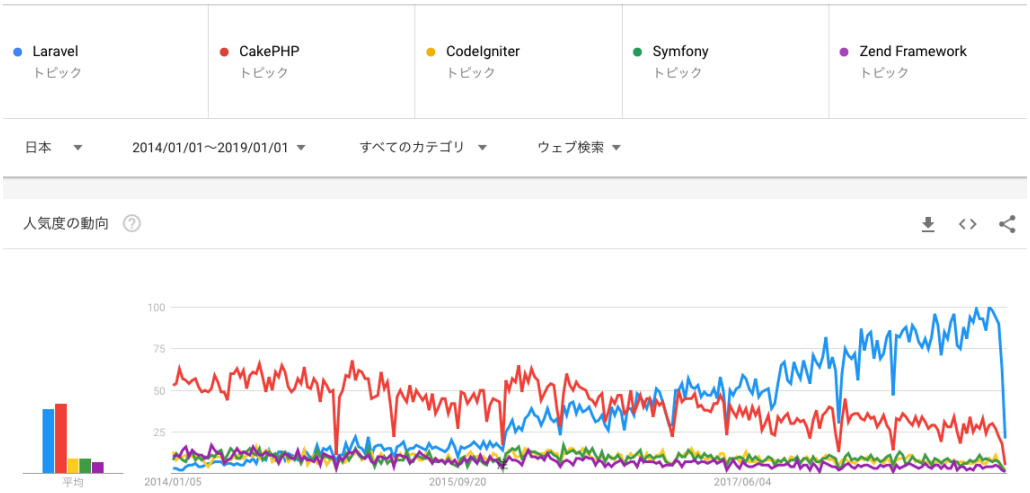

図 2.6: 日本のインターネット (google) 上での PHP フレームワークの検索数

出典 https://trends.google.co.jp

日本国内におけるデータからも，2016 年頃から Laravel が最も検索数の多い CakePHP を抜き，PHP フレームワークの中で最も注目されている Web アプリケーションフレームワークとなっていることがわかる．

2.4.1 Laravel の認証システムについて

Laravel とは，MVC の Web アプリケーション開発用の無料・オープンソースの PHP で書かれた Web アプリケーションフレームワークである．MVC（Model View Controller）とは，ユーザインタフェースをもつアプリケーションソフトウェアを実装するためのデザインパターンである．MVC では，プログラムを 3 つの要素，Model（モデル），View（ビュー），Controller（コントローラ）に分割する．MVC とは，ソフトウェアの設計モデルの一つで，処理の中核を担う『Model』，表示・出力を司る『View』，入力を受け取ってその内容に応じて View と Model を制御する『Controller』の 3 要素の組み合わせでシステムを実装する方式．それぞれの特徴と関係を以下の図 2.7 に示す．

(1) Model(モデル)

Model とは，システムの中でビジネスロジックを担当する，いわばシステムの本体部分にあたるものである．Model は入出力や表示といった処理を行うことはできない．Model とは，

そのアプリケーションが扱う領域のデータと手続きを表現する要素である．また，データの変更を View(ビュー) に通知したり，データベースへのアクセス，レコードの取得と管理を担う要素である．

(2) View(ビュー)

View とは，Model のデータを取り出してユーザが見るのに適した形で表示する要素である．例えば，Web アプリケーションでは HTML 文書を生成して動的にデータを表示するためのコードなどにあたる．

(3) Controller

Controller とはユーザからの入力を Model へのメッセージへと変換して Model に伝える要素である．すなわち，View と Model を制御を行うものである．自分自身では表示を行ったり，ロジックの実行は行わず，View からの入力に応じて，必要なロジックの実行を Model に依頼し，その結果表示を View に依頼する．Model に変更を引き起こす場合もあるが，直接に描画を行ったり，Model の内部データを直接操作したりはしない．

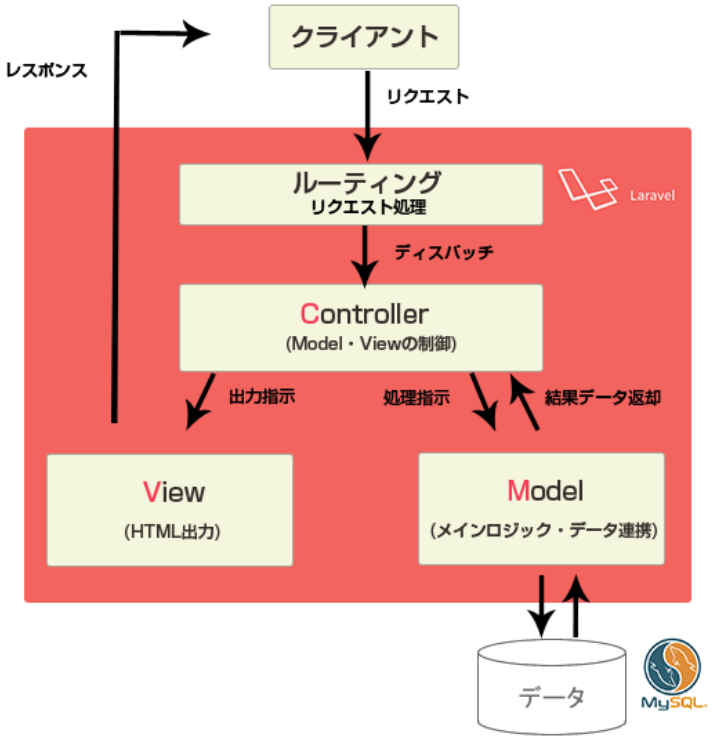

図 2.7: MVC モデルの構造　出典 https://laraweb.net/surrounding/922/

MVC モデルを確立する利点として，このモデルに従うことで，機能ごとの分離が明確になり，それぞれの独立性が確保される．開発においては分業がしやすくなり，各機能の専門家は自分の得意分野の実装に集中することが可能となる．また，コンポーネント間の依存性が最小限に抑えられるため，ほかの部分の変更による影響を受けにくい実装にすることができる．また，ほかのコンポーネントの変更が原因で，複数の担当者が同一のソースに対してメンテナンスを行うような事態を避けることもでき，保守性も確保されるなど，多くのメリットが存在する．また，Laravel の中には既存の認証システムが完備されている．Laravel が持つ MVC モデルの認証システムはユーザのユーザネームとメールアドレス，パスワードを登録し，ログイン時にメールアドレスとパスワードを入力して認証を行う一般的な構造である．システムの流れを以下の図 2.8 および図 2.9 に示す．

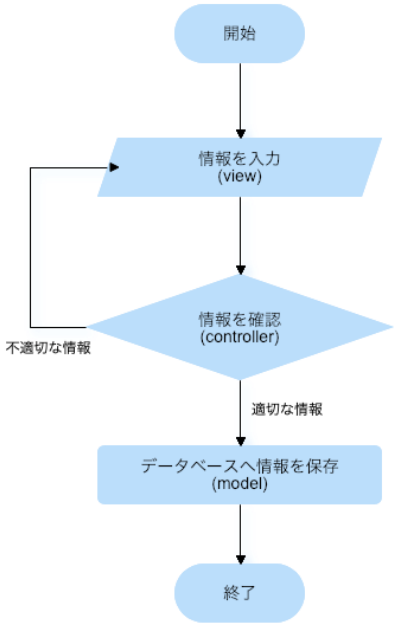

図 2.8: ユーザ情報登録の流れ

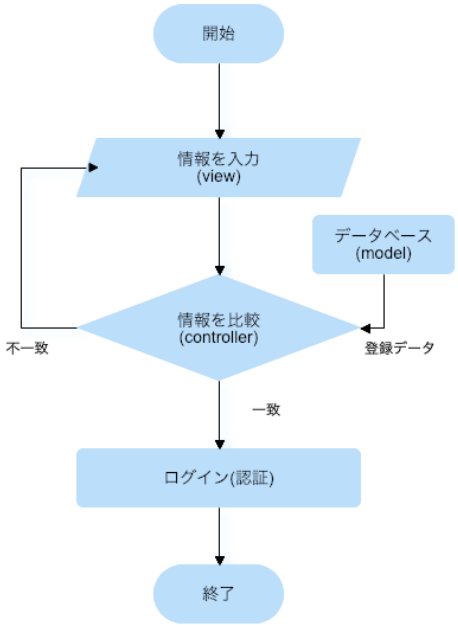

図 2.9: ユーザ情報を用いた認証の流れ

本研究ではこの認証システムの雛形を変更することで，顔画像の比較によりログインを行う仕組みを実現する．

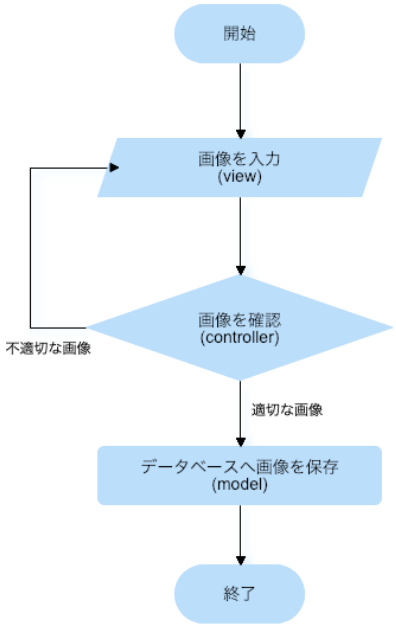

図 2.10: ユーザ顔画像登録の流れ

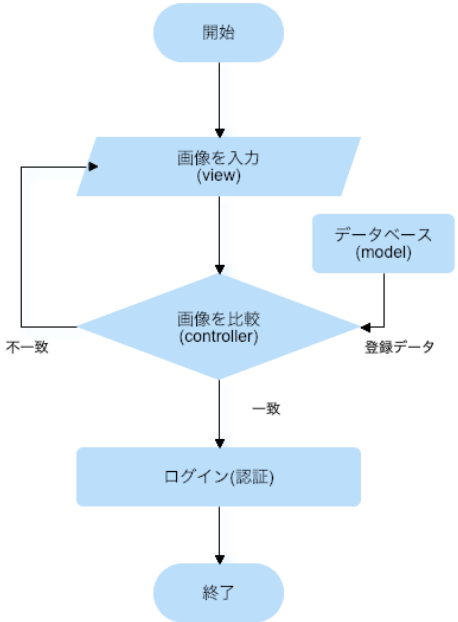

図 2.11: ユーザ顔画像を用いた認証の流れ

また，画像を比較する際の手法として，画像相関法を用いた手法と離散コサイン変換（DCT）を用いた手法とを比較し，検討する．

第 3 章

原理

この章では，２つの画像比較システムおよびその原理について説明をする．

3.1　画像相関法を用いたシステム

画像相関法（正規相互相関法）とは，２つの画像の類似度を計算する方法のことである．類似度を算出するために，２つの画像の輝度値から共分散と標準偏差，さらに相関係数をもとめる必要がある．

3.1.1　共分散

共分散とは 2 種類のデータの関係を示す指標である．共分散を求めるには，2 つの変数の偏差の積の平均を計算する．式は以下の式 (2.1) のようになる．

$$S_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - x_a)(y_i - y_a) \tag{3.1}$$

- n： データの個数
- xi, yi： データの数値
- xa, ya: データの平均値

3.1.2 相関係数

相関係数とは，2 つの確率変数の間にある線形な関係の強弱を測る指標である．相関係数は無次元量で，－ 1 以上 1 以下の実数に値をとる．相関係数が正のとき確率変数には正の相関が，負のとき確率変数には負の相関があるという．また相関係数が 0 のとき確率変数は無相関であるという．また相関係数が 1 に近いほど 2 つの画像は似ていることがわかる．相関係数を求める式を以下の式 (2.2) に示す．

$$r = \frac{\frac{1}{n}\sum_{i=0}^{n} I(x_i - x_a)T(y_i - y_a)}{\sqrt{\frac{1}{n}\sum_{i=0}^{n} I(x_i - x_a)^2}\sqrt{\frac{1}{n}\sum_{i=0}^{n} T(x_i - x_a)^2}} \tag{3.2}$$

- M, N : データの個数
- I, T : それぞれのデータの共分散
- xi, yi : データの数値
- xa, ya: データの平均値

上の式 (2.2) から分かる通り，相関係数はそれぞれの要素の共分散を掛け合わせたものをそれぞれの標準偏差を掛け合わせたもので割ることで求めることができる．

3.2 離散コサイン変換を用いたシステム

3.2.1 顔検出

顔検出から DCT を行うまでの流れをに示す．大まかな流れとしては，まず Web サーバ上でカメラを起動し，顔画像を撮影する．その後，撮影した画像に対し顔検出を行う．さらに，得られた顔の位置座標を使い，元の画像から顔の領域のみをトリミング，リサイズをすることで 64*64pixel の画像とする．その後グレースケールに変換することで入力画像とする．

グレースケール変換後

図 3.1: グレースケール変換のイメージ

3.2.2 離散コサイン変換（DCT）

顔検出を行った後,，顔認識の精度向上のため，DCT を行う．DCT とは，離散信号を周波数領域へ変換する方法の一つである．画像をサンプリングして離散的な信号に変換し，DCT を行った後に符号化を行うことで,，元の信号の大部分を損ねずにデータの容量を減らすことができ，サーバ通信間での実現も高く見込まれる． DCT は離散フーリエ変換と密接に関連している．DCT は分離可能な線形変換の 1 つ，つまり 2 次元変換は，ある次元に沿って 1 次元 DCT を実行した後，もう 1 つの次元で 1 次元 DCT を実行することに相当する．

式（2.3）に入力イメージ f と出力イメージ F に関する 2 次元 DCT の定義を示す．

$$Fi,j = CiCj \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) cos[\frac{\pi(2x+1)i}{2M}] cos[\frac{\pi(2y+1)j}{2N}] \tag{3.3}$$

$$0 \leq i \leq M-1 \tag{3.4}$$

$$0 \leq j \leq N-1 \tag{3.5}$$

$$C_k = \begin{cases} \frac{1}{\sqrt{2}}, (k=0) \\ 1, (k \neq 0) \end{cases} \tag{3.6}$$

M と N はそれぞれ f の行サイズと列サイズである．DCT は変換後の周波数成分が低周波数領域に集中するという特徴があり，DCT による顔画像認識では DCT 係数が個人によって大きく異なることを利用することができる．つまり，図 3.2，図 3.3 に示すように，２次元 DCT 後の低周波数領域に集中した DCT 係数を顔の特徴点として得ることができ，登録した顔画像と別の顔画像を認証システムに用いることが可能となる．

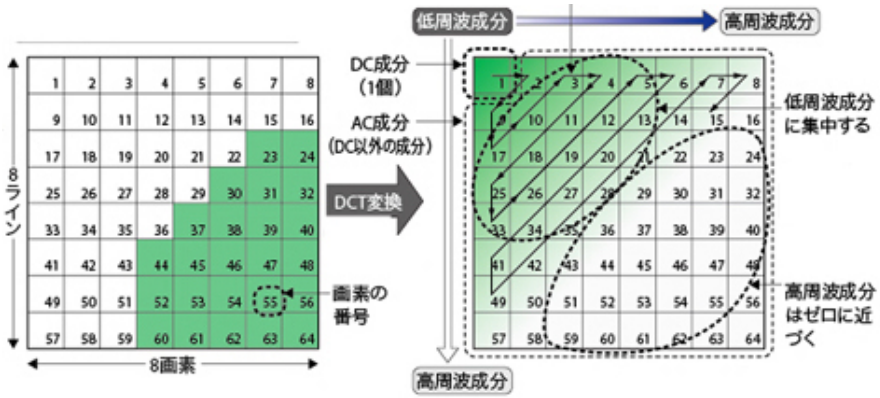

図 3.2: DCT 係数からの特徴抽出のイメージ　出典 https://sgforum.impress.co.jp/

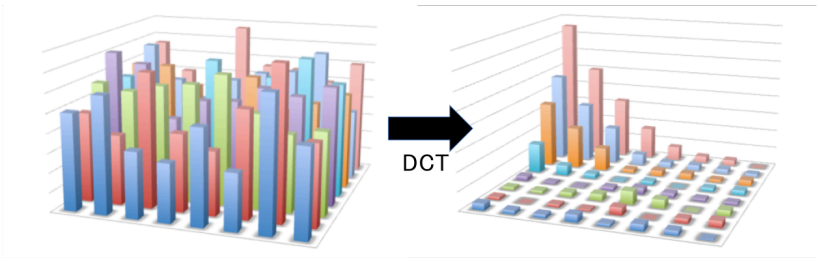

図 3.3: DCT のイメージ

以上，各手法の比較から，本研究では WebRTC による顔検出処理を含む一連の流れとして，離散コサイン変換（DCT）を用いた手法を採用する．

第 4 章

実験

4.1　実験に伴うシステムの前提条件

顔認証システムとは生体認証の 1 つであり，画像から人を識別することができるアプリケーションである．カメラに対して正面に顔を写し, 顔と思われる部分から，あらかじめ登録された情報と比較することで識別を行う．このように顔認証システムは，顔を正面に向けあまり動かさないことを想定しており，正面に向けてない場合は対応できない．そのため，本研究においても同様の想定をし，顔を上下左右に極端に傾けること，カメラから極端に近づいたり離れたりすることがないものとして実験を行う．

4.2　実験方法

本研究における登録用, 認証用画像には，開発した Web アプリケーションフレームワーク上で実際に Web カメラを用いて撮影した顔画像とする．以下はそのカメラの仕様である．

- 使用機器名：Logicool HD Webcam C270m
- 画素数：300 万画素
- 最大解像度：1280*720pixel
- 最大フレームレート：30fps

また，登録用，認証用画像は以下のような条件を満たすものとする．

- 画像中に人の顔が存在している
- カラー画像である

- jpeg 画像である
- 撮影画像の大きさは 480*320 pixel とする
- 前述，システムの前提条件を満たす

前項までに述べた内容で，Web アプリケーションフレームワーク内で動作する顔認証システムを開発し，検証，実験を行った．図 4.1 に本システムの認証の流れを示す．

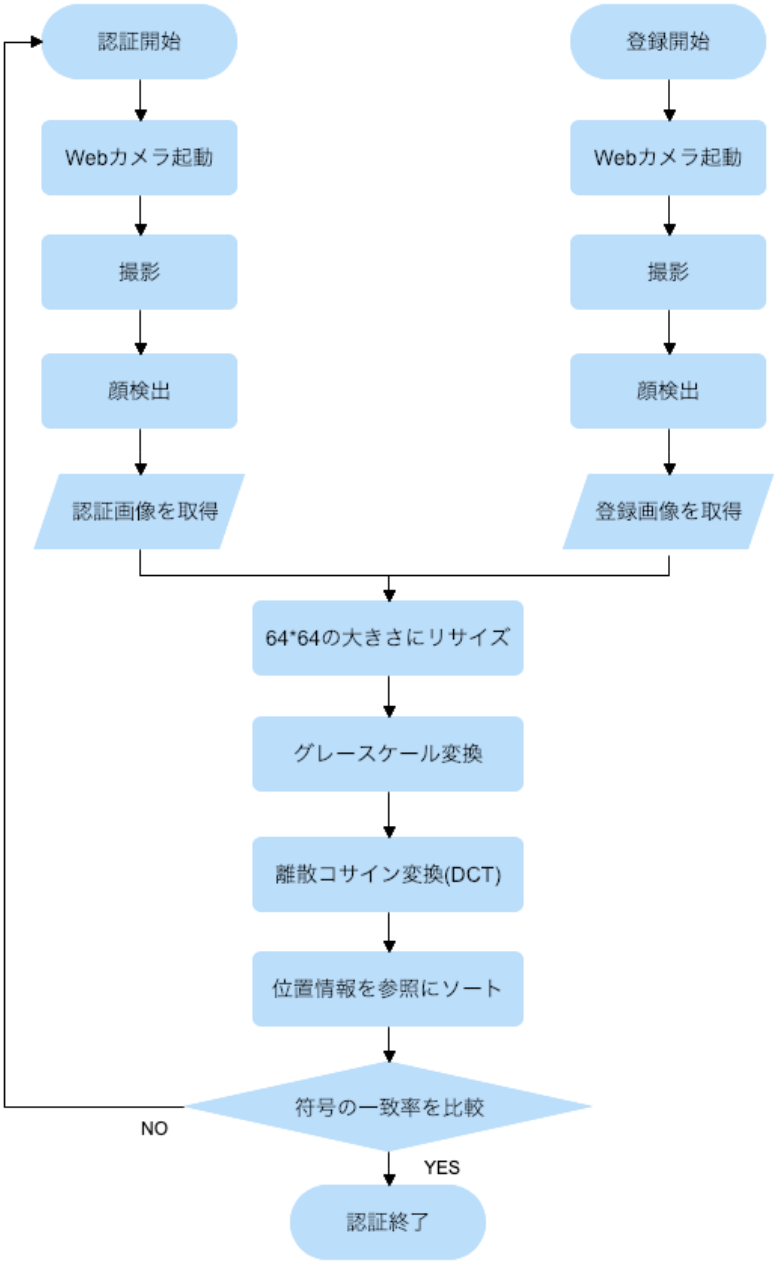

図 4.1: プログラムのフローチャート

Web アプリケーションフレームワーク上でカメラを起動し，画像を取得する． 次に画像内のどこに顔があるかを Shape Detect API を用いた顔検出によって特定し, 得られた顔領域を切り出す．新たに取得した顔領域の画像を 64*64 pixel にリサイズし，認証用画像とする．

ここで，認証用画像とあらかじめ登録された顔画像の 2 枚を入力画像とし，それぞれに対してグレースケール変換，離散コサイン変換 (DCT) を行う．

また認証の方法は，登録用画像の DCT 係数の位置情報を参照し，同じ位置の DCT 係数の符号を比較，その一致率が閾値を超えた場合，DCT 係数から顔画像の特徴情報を抽出し，顔を認識し判別することができたとし，認証を行うように定める．

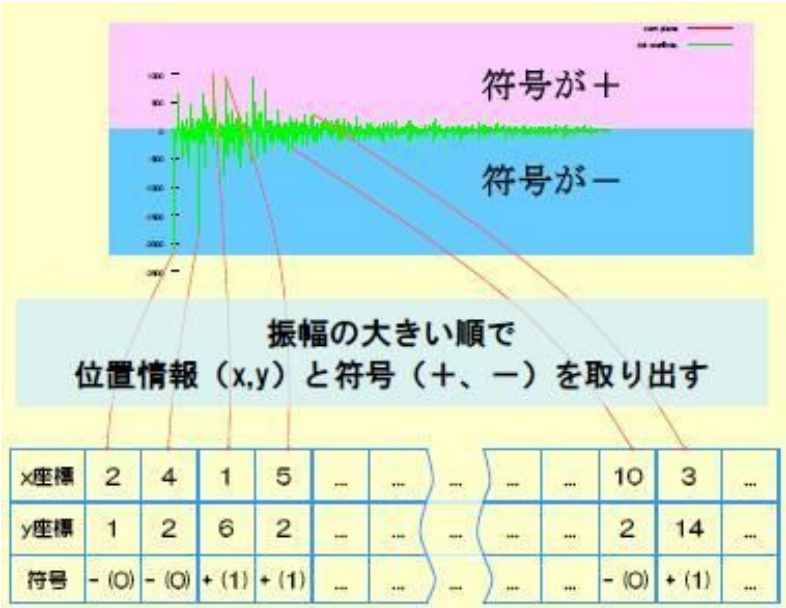

図 4.2: DCT 係数による特徴情報の判別方法

出典 http://www.ccr.kyutech.ac.jp/professors/tobata/t3/t3-2/entry-502.html

後述の実験では，開発した顔認証システムが Web アプリケーション上において適切に動作するため，以下の検証，実験を行う．

- DCT 係数の値条件，抽出範囲の検証
- 認証実験

検証，実験方法として，人物 A,B,C の 3 人の顔画像を 4 枚ずつ，人物 D の顔画像を 1 枚用意する．また，人物 A,B,C の 3 人には各 1 枚ずつの登録画像と各 3 枚ずつの認証画像をそれぞれ用意し，人物 D には 1 枚の登録画像を用意する．人物 A,B,C3 人それぞれの登録画像

に対して，同一人物の認証画像を 3 枚，異なる人物 3 人の登録画像 1 枚の計 3 枚を開発したシステム内で比較し，符号の一致率を検証する．また使用した人物 A,B,C,D の顔画像を以下の図 4.3，図 4.4，図 4.5，図 4.6 に示す．

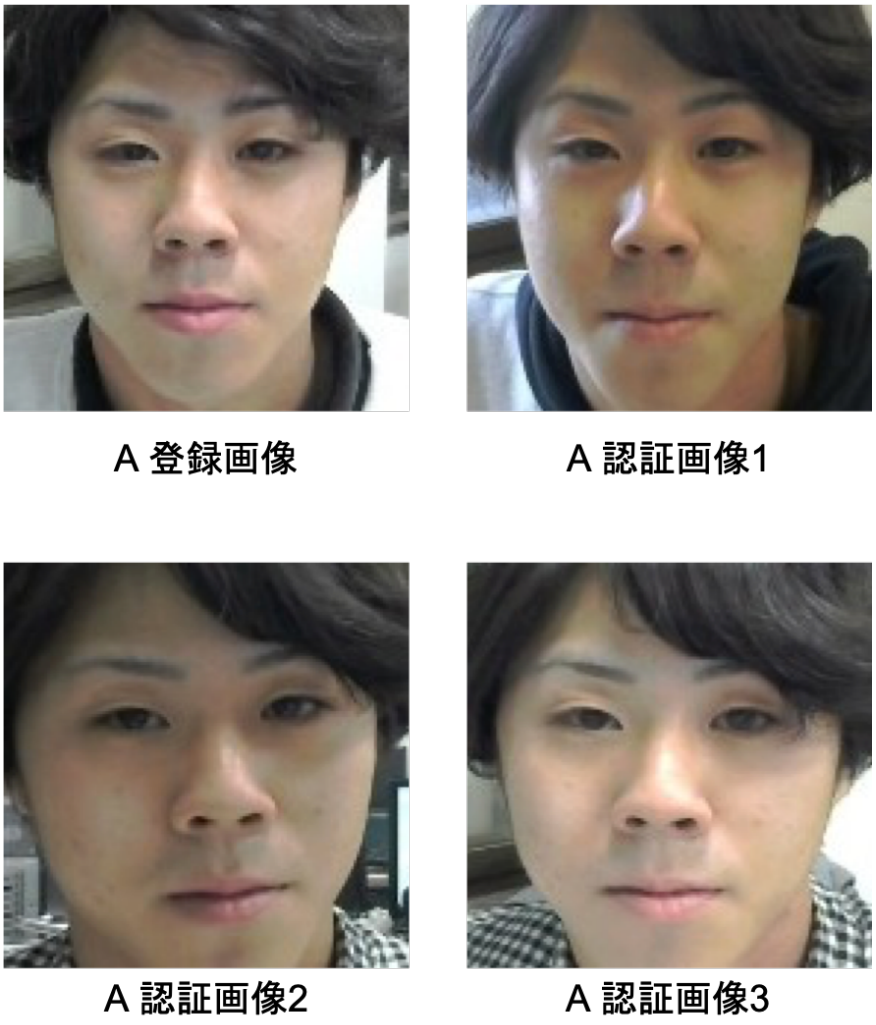

図 4.3: 人物 A の顔画像

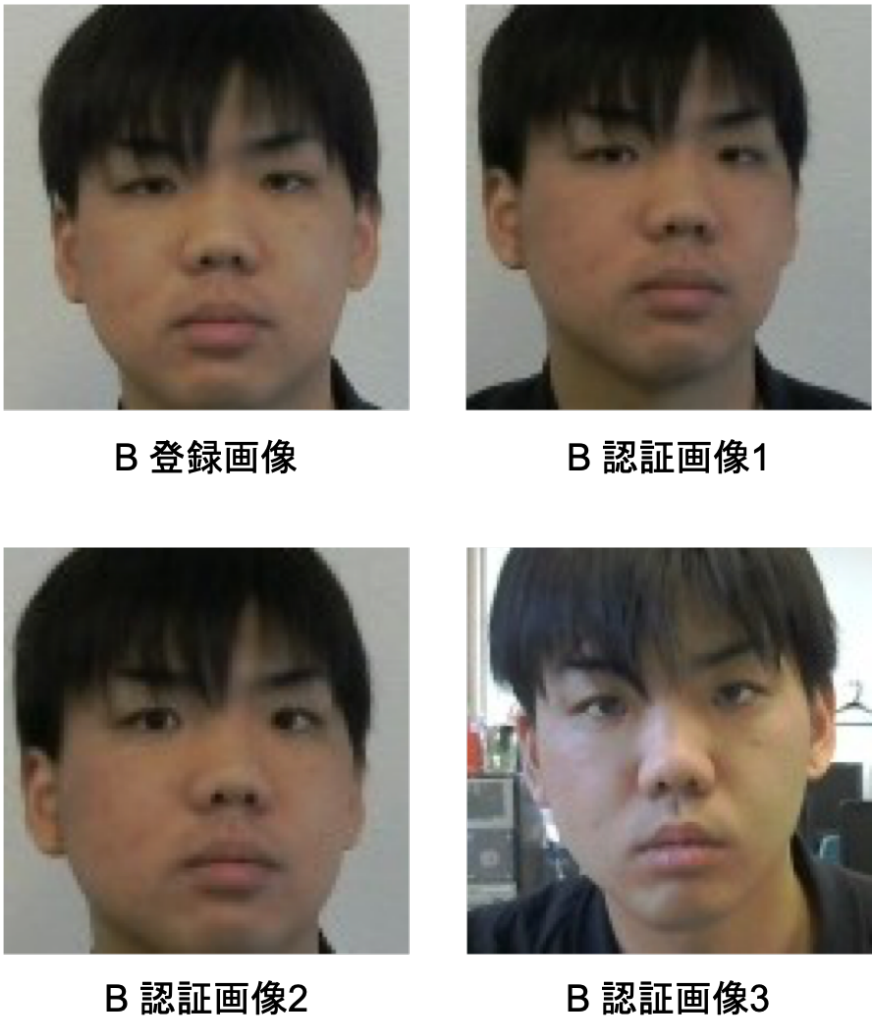

図 4.4: 人物 B の顔画像

図 4.5: 人物 C の顔画像

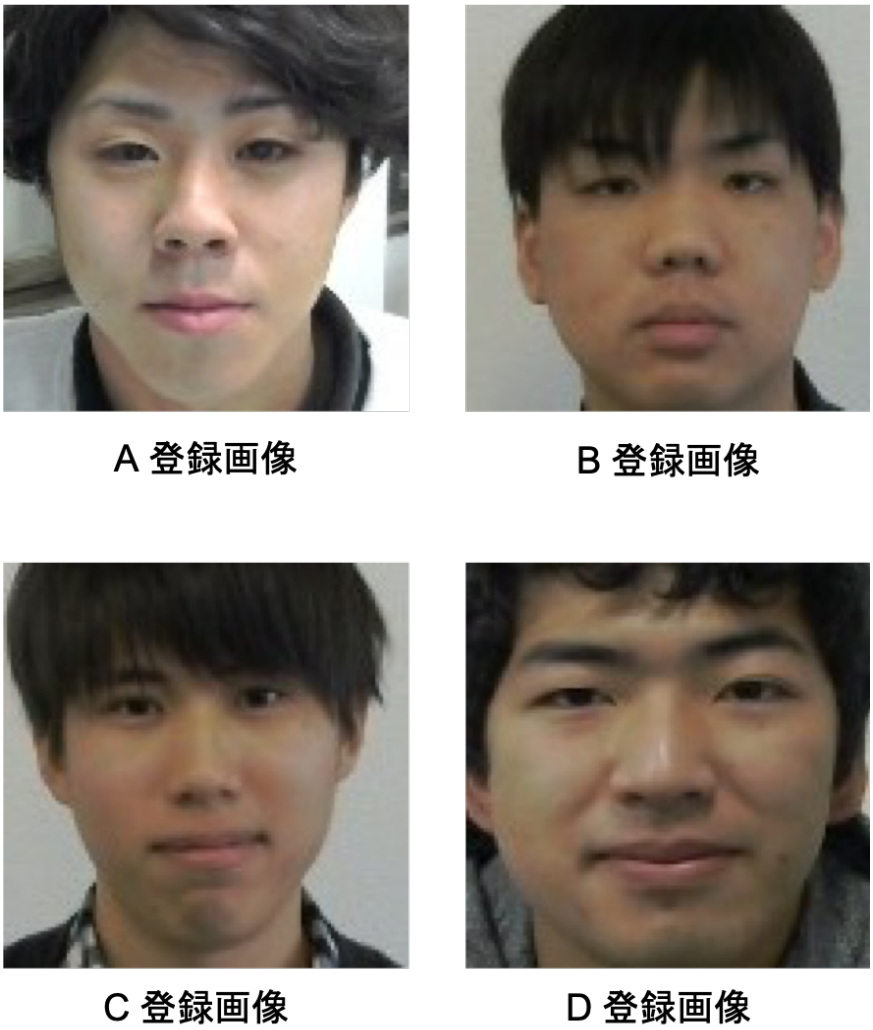

図 4.6: 異なる人物を比較した際に用いた A,B,C,D の各登録顔画像

4.3 値条件，抽出範囲の検証

登録用画像を離散コサイン変換（DCT）することで得られた DCT 係数から，特に条件を定めることなく絶対値の大きい順で 64 点を抽出する．そのうち何点の符号が，認証用画像から得られる同位置の DCT 係数 64 点と一致するかを検証した．その際の，同一人物における符号の一致率および異なる人物における符号の一致率を，比較対象となった DCT 係数の点数に対する一致した点数とし，以下の表 4.1 に示す．

表 4.1: A,B,C に対する DCT 係数の符号の一致率

	認証 1	認証 2	認証 3	一致率 (本人)	A	B	C	D	一致率 (他人)
A	43/64	40/64	43/64	46.88%	×	45/64	45/64	29/64	61.98%
B	35/64	41/64	51/64	66.15%	39/64	×	45/64	29/64	58.85%
C	54/64	51/64	49/64	80.19%	45/64	45/64	×	34/64	64.58%

表 4.1 より，人物 A,B,C それぞれに対して同一人物の認証用画像を入力した際の認証率には大きくばらつきが見られた．また，異なる人物の顔画像を入力とした際の認証率は平均約 60 %となり，とても認証システムとしてレベルを満たす結果が得られなかった．その要因として，本研究では，DCT 係数の符号のプラスとマイナスが一致するかどうかのみで判別を行っているため，50 %ほどは符号が偶然一致してしまうからだと考えられる．そのため，例えば，比較した DCT 係数の点のうち 80 %の符号が一致していたとしても同じ人物であると断定するのは危険であると考えられる．

4.3.1 検証方法

表 4.1 より，DCT 係数の抽出方法や値の大きさによって顔認識のパフォーマンスが変化することが考えられるため，後述の実験では，同一人物に対する符号の一致率が 90 %以上と判断すると同時に，異なる人物に対する符号の一致率が 50 %以下もしくは 50 %に最も近づくような DCT 係数の値の条件および抽出範囲を検証する．

(1) 値条件の検証方法

本研究では登録用画像の DCT 係数の絶対値の大きな 64 点を特徴点として抽出し，比較している．しかし，登録された顔画像には，抽出した 64 点の全てが特徴点に適する大きさを示すとは限らない．そこで，抽出した 64 点のうち特徴点として適する点のみを比較対象とするため，DCT 係数の絶対値を 100 以上，200 以上，300 以上とそれぞれ定め，64 点内に含まれるも条件を満たさないものは比較の対象外とする場合の実験行い，最適な値条件を検証する．

(2) 抽出範囲の検証方法

本研究では登録用画像の DCT 係数の絶対値の大きな 64 点を特徴点として抽出し，比較している．しかし，登録された顔画像から抽出した 64 点には，目の色や髪の色といったように大きな特徴点といえる一方で，比較する際に個人差が生じづらい点も含まれる．そこで，64 点の DCT 係数を抽出する範囲を 1〜64 点目，9〜72 点目，17〜80 点目と 8 点ずつシフトしたそれぞれの場合の実験行い，最適な抽出範囲を検証する．

4.3.2 検証結果

(i) DCT 係数の抽出範囲を 1〜64 点目に設定した際，値条件を 100 以上，200 以上，300 以上と定め，検証を行った．以下の表 4.2，表 4.3，表 4.4 に，それぞれの検証結果を示す．

表 4.2: A,B,C に対する DCT 係数の符号の一致率（抽出範囲:0〜64 点目, 値条件:100 以上）

	認証 1	認証 2	認証 3	一致率 (本人)	A	B	C	D	一致率 (他人)
A	32/41	22/35	23/31	71.70%	×	20/30	21/30	17/35	61.74%
B	15/21	20/28	21/26	74.54%	20/32	×	24/32	14/31	60.89%
C	28/28	30/31	26/32	92.67%	18/27	20/23	×	13/25	68.54%

表 4.3: A,B,C に対する DCT 係数の符号の一致率（抽出範囲:0〜64 点目, 値条件:200 以上）

	認証 1	認証 2	認証 3	一致率 (本人)	A	B	C	D	一致率 (他人)
A	13/18	11/14	17/19	80.01%	×	12/17	13/19	10/18	64.86%
B	12/13	15/17	14/16	89.35%	10/18	×	14/15	6/12	66.30%
C	19/19	19/20	14/15	96.11%	11/15	14/16	×	6/13	69.01%

表 4.4: A,B,C に対する DCT 係数の符号の一致率（抽出範囲:0〜64 点目, 値条件:300 以上）

	認証 1	認証 2	認証 3	一致率 (本人)	A	B	C	D	一致率 (他人)
A	8/11	6/7	12/13	83.58%	×	7/12	8/11	7/10	67.02%
B	8/9	11/11	10/11	93.28%	6/11	×	10/11	4/8	65.15%
C	10/10	12/12	7/8	95.83%	9/11	7/8	×	5/10	73.1%

(ii) DCT 係数の抽出範囲を 9〜72 点目に設定した際，値条件を 100 以上，200 以上，300 以上と定め，検証を行った．以下の表 4.5，表 4.6，表 4.7 に，それぞれの検証結果を示す．

表 4.5: A,B,C に対する DCT 係数の符号の一致率（抽出範囲:9〜72 点目, 値条件:100 以上）

	認証 1	認証 2	認証 3	一致率 (本人)	A	B	C	D	一致率 (他人)
A	29/36	20/28	21/26	77.58%	×	16/24	18/29	13/24	60.97%
B	14/18	28/23	16/21	77.40%	18/28	×	18/23	11/23	62.01%
C	22/23	25/26	19/26	88.29%	17/23	14/18	×	9/17	68.21%

表 4.6: A,B,C に対する DCT 係数の符号の一致率（抽出範囲:9〜72 点目, 値条件:200 以上）

	認証 1	認証 2	認証 3	一致率 (本人)	A	B	C	D	一致率 (他人)
A	10/12	9/10	13/13	91.11%	×	7/11	8/14	6/11	58.44%
B	8/9	13/14	11/12	91.14%	7/13	×	9/14	4/7	58.42%
C	12/13	14/14	9/9	97.44%	6/10	7/11	×	3/6	57.88%

表 4.7: A,B,C に対する DCT 係数の符号の一致率（抽出範囲:9〜72 点目, 値条件:300 以上）

	認証 1	認証 2	認証 3	一致率 (本人)	A	B	C	D	一致率 (他人)
A	6/8	3/3	7/7	91.67%	×	4/8	5/7	3/4	70.24%
B	5/6	8/8	6/7	89.68%	5/9	×	6/7	3/4	72.09%
C	5/6	8/8	3/3	94.44%	4/6	2/2	×	2/3	77.78%

(iii) DCT 係数の抽出範囲を 17〜80 点目に設定した際，値条件を 100 以上，200 以上，300 以上と定め，検証を行った．以下の表 4.8，表 4.9，表 4.10 に，それぞれの検証結果を示す．

表 4.8: A,B,C に対する DCT 係数の符号の一致率（抽出範囲:17〜80 点目, 値条件:100 以上）

	認証 1	認証 2	認証 3	一致率 (本人)	A	B	C	D	一致率 (他人)
A	28/36	22/28	20/25	77.88%	×	16/24	18/29	13/24	60.97%
B	14/16	18/21	15/20	82.74%	18/26	×	18/23	9/21	63.44%
C	21/22	25/26	19/26	88.23%	17/23	14/18	×	9/16	69.31%

表 4.9: A,B,C に対する DCT 係数の符号の一致率（抽出範囲:17〜80 点目, 値条件:200 以上）

	認証 1	認証 2	認証 3	一致率 (本人)	A	B	C	D	一致率 (他人)
A	10/12	9/9	13/13	94.44%	×	7/10	8/14	5/8	63.21%
B	8/9	13/13	9/10	92.97%	7/12	×	9/10	4/7	68.49%
C	12/13	14/14	9/9	97.44%	6/10	7//9	×	3/6	62.59%

表 4.10: A,B,C に対する DCT 係数の符号の一致率（抽出範囲:17〜80 点目, 値条件:300 以上）

	認証 1	認証 2	認証 3	一致率 (本人)	A	B	C	D	一致率 (他人)
A	6/8	3/3	7/7	91.67%	×	4/8	5/7	3/4	70.24%
B	5/8	8/8	6/7	89.68%	5/9	×	6/7	3/4	72.09%
C	5/6	8/8	3/3	94.44%	4/6	2/2	×	2/3	77.78%

4.4　認証実験

前述の検証結果 (i)，(ii)，(iii) より，登録人物と同一人物への符号の一致率および異なる人物への符号の一致率の，各条件下における実験の結果に共通した傾向がみられ，符号の一致率は DCT 係数の値の大きさと抽出範囲に大きく影響していることがわかる．

そこで後述の実験では，DCT 係数の符号の比較結果が，同一人物に対する符号の一致率が 90 %を超え，かつ異なる人物に対する符号の一致率が最も 50 %に近づく結果となった以下の条件に定め，本研究の顔認証システムの精度を検証する．

- 値条件：絶対値 200 以上
- 抽出範囲：9〜72 点目
- 閾値：90 %

4.4.1　認証実験の方法

登録用画像を離散コサイン変換（DCT）することで得られた DCT 係数から，絶対値の大きい順でソートしたものから 9〜72 点目の計 64 点を抽出する．そのうち，絶対値の大きさが 200 以上を示す値の何点が，認証用画像から得られる同位置の DCT 係数と符号が一致するかを比較し，顔認証を行う．認証の閾値を 90 %とし，図 4.6 の人物 A,B,C の各登録顔画像に対して，人物 A,B,C,D の顔画像を 50 枚ずつ認証する．その際の認証枚数および本人拒否率 (FRR)，他人受入率 (FAR) を検証する．

4.4.2　認証実験の結果

(a)　人物 A の登録顔画像に対して，A,B,C,D の顔画像を各 50 枚ずつ認証した際の認証枚数およびそこから算出される本人拒否率 (FRR)，他人受入率 (FAR) を表 4.11 に示す.

表 4.11: 人物 A の登録顔画像に対する認証枚数 (50 枚中)

	A [枚]	B [枚]	C[枚]	D[枚]	FRR [%]	FAR [%]
A	47	2	3	0	6.00	3.33

図 4.7: 人物 A の登録顔画像に対する誤認識顔画像

(b)　人物 B の登録顔画像に対して，A,B,C,D の顔画像を各 50 枚ずつ認証した際の認証枚数およびそこから算出される本人拒否率 (FRR)，他人受入率 (FAR) を表 4.12 に示す.

表 4.12: 人物 B の登録顔画像に対する認証枚数 (50 枚中)

	A [枚]	B [枚]	C[枚]	D[枚]	FRR [%]	FAR [%]
B	1	48	3	4	4.00	5.33

図 4.8: 人物 B の登録顔画像に対する誤認識顔画像

(c) 人物 C の登録顔画像に対して，A,B,C,D の顔画像を各 50 枚ずつ認証した際の認証枚数およびそこから算出される本人拒否率 (FRR)，他人受入率 (FAR) を表 4.13 に示す.

表 4.13: 人物 C の登録顔画像に対する認証枚数 (50 枚中)

	A [枚]	B [枚]	C[枚]	D[枚]	FRR [%]	FAR [%]
C	2	1	49	3	2.00	4.00

図 4.9: 人物 C の登録顔画像に対する誤認識顔画像

認証実験では，表 4.11，表 4.12，表 4.13 より，本人拒否率 (FRR) が 4.00 %，他人受入率 (FAR) が 4.22 %の結果が得られた．また同様の実験を，閾値を 1 %ずつ上昇させ，行う．その際に算出された本人拒否率 (FRR) および他人受入率 (FAR) の推移を以下の図 4.10 に示す．

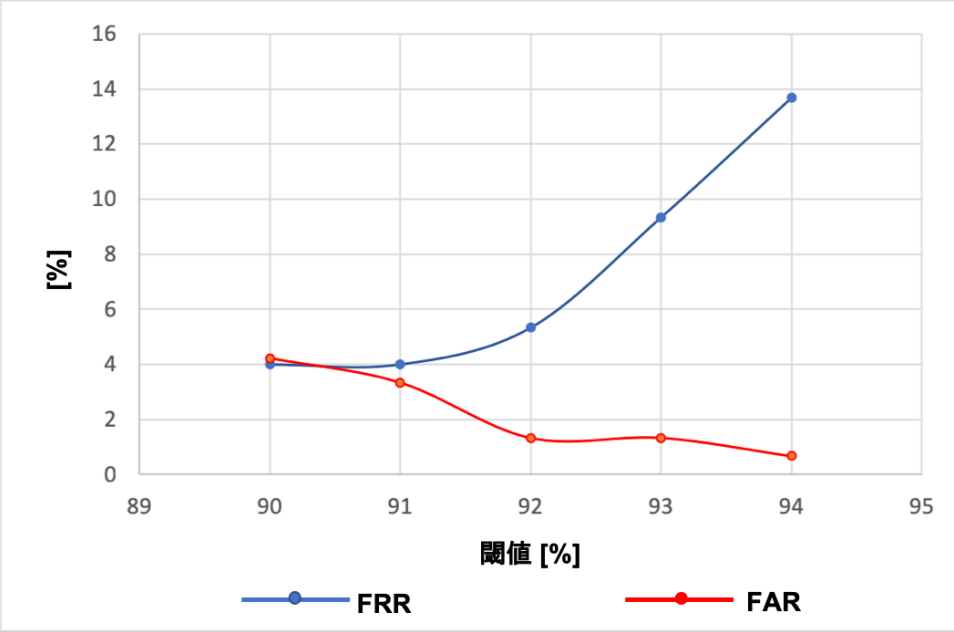

図 4.10: 閾値の変化に対する本人拒否率 (FRR) および他人受入率 (FAR) の推移

4.5 考察

DCT 係数の値条件および抽出範囲に関する検証の結果より，DCT 係数の符号の一致率は値の大きさと抽出範囲に大きく影響することがわかる．まず，DCT 係数の示す値の大きさについて，同抽出範囲内において値条件ををそれぞれ絶対値 100 ずつ上昇させると，同一人物および異なる人物に対する比較点の数が減少していることがわかる．これにより，抽出した 64 点内に含まれる個人の顔を判別するに値しないほどの点を比較対象外にすることが可能となった．このことは検証結果 (i)，(ii)，(iii) いずれの抽出範囲において，値条件を 100 から 200 に上昇させた場合では，異なる人物への符号の一致率に大きな変化がない一方で，同一人物への符号の一致率のみが上昇していることからわかる．また，DCT 係数を抽出する範囲について，同値条件において抽出範囲を 8 点ずつシフトしていくと，同一人物および異なる人物に対する比較点の数がいずれも減少し，符号の一致率が上昇していることがわかる．これは，本システムが DCT 係数を降順でソートしたものから範囲を指定し，点を抽出しているため，ソートしていくにつれて設定した値条件を満たす点が減少するからである．その結果，DCT 係数の絶対値が大きく，特徴点といえる一方で，他人との差異が出ない点を比較対象から外し，特徴点として適切な点を比較することができる．しかし，検証結果 (iii) のように，抽出範囲を極端にソートした場合，比較対象となる DCT 係数の点が過少し，同一人物への符号の一致率および異なる人物への符号の一致率がいずれも上昇してしまい，個人の顔の判別が困難になったと考えられる．

検証に基づき DCT 係数の値条件および抽出範囲を設定し，認証の閾値を 90 %とした実験では，他人受入率 (FAR) が本人拒否率 (FRR) を上回る結果となった．しかし，認証システムにおいてセキュリティ面を考慮すると，まず何よりも誤って他人を受け入れないことが重要である．そこで，閾値を 1 %ずつ上昇させ行った認証実験では，本顔認証システムは閾値 91 %に設定した際，他人受入率 (FAR) が本人拒否率 (FRR) を下回ることが図 4.10 よりわかる．また同様の認証実験を閾値 94 %に設定し，行った際には，他人受入率 (FAR) が 1 %以下となることがわかった．しかし一方で，他人受入率 (FAR) が低下するのに伴い，本人拒否率 (FRR) が増加した．つまり登録人物と同一人物の顔画像にも関わらず，認証されないということが多くなり，利便性が失われて

しまった．これらのことから，安全性を考慮するため本人拒否率 (FRR) が多少大きくなってもいいので他人受入率 (FAR) を小さくしながらも，誤って正規ユーザをログイン拒否することのないよう，他人受入率 (FAR) と本人拒否率 (FRR)，つまり安全性と利便性のトレードオフが重要になると考えられる．

第 5 章

結論

本研究では無料のオープンソースのフレームワークである Laravel を用いて顔画像を比較する手法による新たな認証システムを作成した．その結果，フレームワークの中でも画像処理を行い，認証が行えることがわかった．また PHP は，C++，Python のように openCV のような画像処理ライブラリが使用できる言語ではないため，画像処理を行うプログラムを省略できない点が問題であったが，GD 関数を用いることで代用でき，離散コサイン変換 (DCT) による基本的な画像処理は PHP でも実装が可能であることが確認できた．

また，本研究で開発した認証システムでは，DCT 係数の符号のみで顔認証を行い，さらに抽出する点の大きさや抽出点の範囲によって，そのパフォーマンスがどのように変化するかの検討を行った．その結果，DCT 係数の符号のみで個人の顔を判別することが可能であることがわかったとともに，Web サーバ上で顔認証システムを実現することが可能であることも判明した．

ただし，本実験で認証の際に用いる画像は意識的に顔に変化を与えようとしていないため，実際に対象人物の協力のもとカメラに対し正面を向いて静止してもらえれば認証に失敗することはほとんどないと考えられるが，本実験の前提条件に満たない場合の認証性能はまだ課題がみられる．また，離散コサイン変換 (DCT) の計算処理および DCT 係数のソートや比較を行う認証速度が，通常のシステムと比較して約 1/3 ほどとなってしまった．そこで今後，DCT 後の特徴点の抽出方法や計算方法の工夫することで，本実験の前提条件外での認証率の向上が期待できるうえに，サーバ間の認証プログラムの処理速度の高速化も期待できる．また機械学習を導入することで，DCT 係数を何点取るのか，どの場所を取るのかをコンピュータが自動的に判断してくれるようにすれば，さらなる効果が期待でき，Web サーバ上で行う顔認証シス

テムとしてはかなり有用なものになると考えられる.

謝辞

本論文は筆者が九州工業大学電気電子工学科電子工学コースに在籍中の研究成果をまとめたものである．同学科准教授の張力峰先生には指導教員として本研究の実施の機会を与えていただき，その遂行にあたって終始，ご指導をいただいた．ここに深謝の意を表する．同学科教授芹川聖一先生，同学科准教授山脇彰先生，同学科助教授楊世淵先生には研究遂行にあたり日頃から数多くのご助言，ご指導をいただいた．ここに深謝の意を表する．そして張研究室，芹川研究室，山脇研究室の皆様には研究遂行にあたり多くのご協力をいただいた．ここに深謝の意を表する．

参考文献

[1] 図録　日本の高齢化の推移と将来推計,

<https://www8.cao.go.jp/index.html>

[2] 図録　国内利用者数および人口普及率の推移,

<http://www.soumu.go.jp/johotsusintokei/statistics/statistics05.html>

[3] 図録　国内世代別インターネット利用率の推移,

<http://www.soumu.go.jp/johotsusintokei/statistics/statistics05.html>

[4] 図録　Web アプリケーションの構造,

<http://www.claire.co.jp/service/webappli>

[5] 図録　Java フレームワーク,

<https://furien.jp/columns/46/>

[6] 図録　PHP フレームワーク,

<https://furien.jp/columns/46/>

[7] 図録　Ruby フレームワーク,

<https://furien.jp/columns/46/>

[8] 図録　世界のインターネット (google) 上での PHP フレームワークの検索数,

<https://trends.google.co.jp>

[9] 図録　日本のインターネット (google) 上での PHP フレームワークの検索数,

<https://trends.google.co.jp>

[10] 図録　MVC モデルの構造,

<https://laraweb.net/surrounding/922/>

[11] 図録　DCT 係数からの特徴抽出のイメージ,

<https://sgforum.impress.co.jp/>

[12] 酒井幸市 「画像処理とパターン認識入門: 基礎から VC/VC++.NET によるプロジェク

ト作成まで」, 森北出版, 2006, pp.101-112

[13] 長橋宏 「画像解析論（３）」,
<http://www.isl.titech.ac.jp/nagahashilab/member/longb/imageanalysis/LectureNotes/ImageAnalysis03.pdf>

[14] Robin Nixon, 永井勝則 「初めての PHP，MySQL，JavaScript & CSS 第 2 班」, 株式会社オライリージャパン, 2013

[15] 掌田津耶乃 「PHP フレームワーク Laravel 入門」, 株式会社秀和システム, 2017

付録 A

A.1　本研究で使用したプログラム

A.1.1　Web カメラを用いてサーバ上で顔画像を取得するプログラム

```
<!DOCTYPE html>
<html>
<head>
<meta content="width=device-width initial-scale=1.0 minimum-scale=1.0
maximum-scale=1.0 user-scalable=no" name="viewport">
<meta http-equiv="Content-type" content="text/html; charset=utf-8">
<link rel="shortcut icon"
href="https://lightbox.sakura.ne.jp/homepage/WinOfSql.ico">
</head>
<body>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/2.1.3/toastr.min.css">
<script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/toastr.js/2.1.3/toastr.min.js">
</script>
<style>
input {
```

```
font-size: 20px;
}
#camera {
width: 400px;
height: 300px;
}
#canvas,#camera {
border: 1px solid #000;
}
</style>
<input id="copy" type="button" value="copy">
<input id="save" type="button" value="save">
<br>
<br>
<video id="camera" autoplay></video>
<canvas id="canvas" width="400" height="300"></canvas>
<script>
jQuery.isMobile =
(/android|webos|iphone|ipad|ipod|blackberry|iemobile|opera mini/i.test(navigator.use
toastr.options.positionClass = "toast-top-center";
if ( $.isMobile ) {
$("#camera").css("width","100%");
$("#canvas").css("width","100%");
}
// カメラ用 video 要素 (DOM オブジェクト)
var camera;
// 静止画用 canvas 要素 (DOM オブジェクト)
var cancas;
check();
```

```
// ************************************
// Canvas へコピー
// ************************************
$("#copy").on( "click", function(){
camera = $("#camera").get(0);
canvas = $("#canvas").get(0);
var ctx = canvas.getContext('2d');
// カメラから キャンバスに静止画を描く
ctx.drawImage(camera, 0, 0, 400, 300);
});
// ************************************
// Canvas の画像を保存
// ************************************
$("#save").on( "click", function(){
// IE の場合
if ( typeof(MSBlobBuilder) != "undefined" ) {
var jpeg = canvas.toDataURL("image/jpeg")       // JPEG
var bin = atob(jpeg.split(',')[1]);
var buffer = new Uint8Array(bin.length);
for (var i = 0; i < bin.length; i++) {
buffer[i] = bin.charCodeAt(i);
}
var blob = new Blob([buffer.buffer], {type: "image/jpeg"});
navigator.msSaveBlob(blob, "canvas.jpg" );
}
else {
var jpeg = canvas.toDataURL("image/jpeg")       // JPEG
var download = $("<a></a>").appendTo("body").css("display","none");
download.prop({"href" : jpeg, "download": "canvas.jpg" });
```

```
download.get(0).click();
download.remove();
}
});
// ************************************
// navigator.getUserMedia チェック
// ************************************
function check() {
if ( !navigator.mediaDevices ) {
var api = [
"webkitGetUserMedia", "mozGetUserMedia","msGetUserMedia"
]
$.each(api,function(idx){
if (navigator.getUserMedia = navigator.getUserMedia || navigator[api[idx]]) {
return false;
}
});
if ( !navigator.getUserMedia ) {
error("WebRTC を使用できません");
return;
}
}
// WEB カメラの初期化
init();
}
// ************************************
// カメラストリーム
// ************************************
function init() {
```

```
camera = $("#camera").get(0);
if ( navigator.mediaDevices ) {
console.log("navigator.mediaDevices.getUserMedia");
navigator.mediaDevices.getUserMedia({video: true})
.then(function(stream){
camera.src = window.URL.createObjectURL(stream);
})
.catch(function(err){
error(err.name);
});     }
else {
console.log("navigator.getUserMedia");
navigator.getUserMedia(
{video: true},
function(stream) {
camera.src = window.URL.createObjectURL(stream);
},
function(err){
error(err.name);
}
);
}
}
</script>
</body>
</html>
```

A.1.2 Web サーバ上で画像から顔領域を検出するプログラム

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
</head>
<body>
   <input type="file" id="target" accept="image/*">
   <br>
   <img id="image" style="display:none">
   <script>
         const target = document.getElementById('target');
      target.addEventListener('change', function (e) {
          const file = e.target.files[0]
          const reader = new FileReader();
          reader.onload = function (e) {
              const img = document.getElementById("image")
              img.src = e.target.result;
          }
          reader.readAsDataURL(file);
      }, false);
   </script>

    <canvas id="canvas2"></canvas>
    <script>
        const image = document.getElementById('image');
        const canvas2 = document.getElementById('canvas2');
        const ctx2 = canvas2.getContext('2d');
```

```
        let scale = 1;
        image.onload = function() {
           canvas2.width = image.width;   //Canvas を画像と同じ大きさにする
           canvas2.height = image.height; //Canvas を画像と同じ大きさにする
            ctx2.drawImage(image,
                0, 0, image.width, image.height,
                0, 0, canvas2.width, canvas2.height);
            scale = canvas2.width / image.width;
            detect(); //追加：読み込み完了後に写真に四角い線を描画する
        };

        function detect() {
            if (window.FaceDetector == undefined) {
                console.error('Face Detection not supported');
                return;
            }
            const faceDetector = new FaceDetector();
            faceDetector.detect(image)
                .then(faces => {
                    ctx2.lineWidth = 2;
                    ctx2.strokeStyle = 'red';
                    for (let face of faces) {
                        ctx2.rect(
Math.floor(face.boundingBox.x * scale),
                            Math.floor(face.boundingBox.y * scale),
                            Math.floor(face.boundingBox.width * scale),
                            Math.floor(face.boundingBox.height * scale));
                        ctx2.stroke();
                        console.log(face);
```

```
                    }
                })
                .catch((e) => {
                    console.error("Boo, Face Detection failed: " + e);
                });
        }
    </script>
</body>
</html>
```

A.1.3 DCT を行い，顔の特徴点の一致率を測定するプログラム

```
<?php
   public function  initCoefficients()//二次元 DCT 計算式の固有値
   {
      for ($i=1;$i<64;$i++)
      {
         $c[$i]=1;
      }
      $c[0]=1/sqrt(2.0);
   }
   public function applyDCT($color)//二次元 DCT 計算式
   {
      $c=$this->initCoefficients();
      $N=64;//正方形化後の長さ
      $sum=0;
      for ($u=0;$u<$N;$u++) {
         for ($v=0;$v<$N;$v++) {
            for ($i=0;$i<$N;$i++) {
               for ($j=0;$j<$N;$j++) {
                  $sum += ($c[$i]*$c[$j]) * cos(((2*$i+1)*$u*pi()/(2.0*$N)))
                          * cos(((2*$j+1)*$v*pi()/(2.0*$N))) * ($color[$i*$N+$j]);
               }
            }
            $sum *=sqrt(2/$N)*sqrt(2/$N);
            $F[$u][$v] = $sum;
         }
      }
      return $F;
```

```
}

public function array_associative($array)//連想配列化
{
   for($i = 0; $i < 64*64; $i++)
   {
      $associative_arr[$i] = array(
         'id' => $i,
         'value' => $array[$i]
      );
   }
   return $associative_arr;
}

public function array_valSort($array)//value 基準で並び変え
{
   foreach ((array) $array as $key => $value)
   {
      $sort[$key] = abs($value['value']);//'value' の絶対値で並び変え
   }
   array_multisort($sort, SORT_DESC, $array);//ソートし，配列を上書き
   return $array;
}

public function array_idSort($array)//id 基準で並び変え
{
   foreach ((array) $array as $key => $id)
   {
      $sort[$key] = abs($id['id']);
```

```
    }
    array_multisort($sort, SORT_ASC, $array);
    return $array;
}

public function  array_authRate($array1, $array2)//認証率を計算
{
    $sign = 0;
    $count = 0;
    for ($i=7; $i<70; $i++)
    {
        if(abs($array1[$i])>200 && abs($array2[$i])>200)//絶対値 200 以上
        {
            $count += 1;
            if($array1[$i] * $array2[$i] > 0 )//符号が一致したらカウント
            {
                $sign += 1;
            }
        }
    }
    $rate = round($sign/$count, 4) * 100;
    return $rate;
}

public function calculateCorr($im1, $im2)
{
    for($j = 0; $j < 64; $j++)//$im をそれぞれグレースケール化
    {
        for($i = 0; $i < 64; $i++)
```

```
        {
            $rgb1 = imagecolorat($im1, $i , $j);
            $rgb2 = imagecolorat($im2, $i, $j);
            $count = $j*64 + $i;//1 行に並べた時の順番
            $r1[$count] = ($rgb1 >> 16) & 0xFF;
            $g1[$count] = ($rgb1 >> 8) & 0xFF;
            $b1[$count] = $rgb1 & 0xFF;
            $glay1[$count] = intval(
                ($r1[$count]*0.33 + $g1[$count]*0.34 + $b1[$count]*0.33)
                                    );
            $r2[$count] = ($rgb2 >> 16) & 0xFF;
            $g2[$count] = ($rgb2 >> 8) & 0xFF;
            $b2[$count] = $rgb2 & 0xFF;
            $glay2[$count] = intval(
                ($r2[$count]*0.33 + $g2[$count]*0.34 + $b2[$count]*0.33)
                                    );
        }
    }
    $DCTed_G1 = $this->applyDCT($glay1);
    $DCTed_G2 = $this->applyDCT($glay2);

    $flat_G1 = array_flatten($DCTed_G1);
    $flat_G2 = array_flatten($DCTed_G2);

//連想配列化
    $associative_G1 = $this->array_associative($flat_G1);
    $associative_G2 = $this->array_associative($flat_G2);

//絶対値 value の降順に並び変え
```

```
        $valueSorted_G1 = $this->array_valSort($associative_G1);
        $valueSorted_G2 = $this->array_valSort($associative_G2);

//絶対値 value が大きいものを 256 個を抽出
        $valueLarge_G1 = array_slice($valueSorted_G1, 0, 16*16);
        $valueLarge_G2 = array_slice($valueSorted_G2, 0, 16*16);
        //dd($valueLarge_G2);

//$valueLarge_G1 と同 id のものだけを$associative_G2 から抽出
        $idSame_G2 = array_uintersect(
            $valueLarge_G2, $valueLarge_G1, function ($arr1, $arr2)
            {
                $result2 = $arr1['id'] - $arr2['id'];
                return $result2;
            }
        );

//$valueLarge_G1 と同 id のものだけを$associative_G2 から抽出
        $idSame_G1 = array_uintersect(
            $valueLarge_G1, $valueLarge_G2, function ($arr1, $arr2)
            {
                $result1 = $arr2['id'] - $arr1['id'];
                return $result1;
            }
        );
        $idSorted_G1 = $this->array_idSort($idSame_G1);
        $idSorted_G2 = $this->array_idSort($idSame_G2);

        $valueOnly_G1 = array_column($idSorted_G1, 'value');
```

```
        $valueOnly_G2 = array_column($idSorted_G2, 'value');

        $authRate_G
            = $this->array_authRate($valueOnly_G1, $valueOnly_G2);

        if($authRate_G > 90) return [true, $authRate_G];
        else return false;
    }

    public function returnCocorrelationResult($valuesA, $valuesB)
    {
        return $this->calculateCorr($valuesA, $valuesB);
    }
}
```