一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
KBSE2018-28,SC2018-23 (2018-11)

様々なアプリと連携可能なリアルタイム顔識別デバイスの開発

平山　孝輔†　佐伯　幸郎†　中村　匡秀†,††

† 神戸大学　〒657–8501 神戸市灘区六甲台町 1–1
†† 理化学研究所・革新知能統合研究センター　〒103–0027 東京都中央区日本橋 1–4–1
E-mail: †hirayama@ws.cs.kobe-u.ac.jp, ††sachio@carp.kobe-u.ac.jp, †††masa-n@cs.kobe-u.ac.jp

あらまし　近年，画像認識技術を用いて顔を識別する機能を使ったアプリケーションが開発されている．しかしながら，顔識別機能をアプリケーションに実装する際に，機能の再利用性の低さや，カメラやデバイスの共有性の欠如により，開発者に負担が生じる．そこで本研究では，リアルタイム顔識別デバイスである「顔識別センサボックス」を提案する．提案デバイスは，コグニティブ API を用いた顔識別を行い，その結果を Publish/Subscribe 型のメッセージングモデルを用いて，非同期にアプリケーションに通知する．顔識別センサボックスを利用することで，複数のアプリケーション間で顔識別機能の共有ができるため，アプリケーションの設計が容易になる．
キーワード　顔識別，コグニティブ API，Pub/Sub，IoT，エッジコンピューティング

Developing Real-time Face Identification Device Composable with Distributed Applications

Kosuke HIRAYAMA†, Sachio SAIKI†, and Masahide NAKAMURA†,††

† Kobe University　Rokkodai-cho 1–1，Nada-ku，Kobe，Hyogo，657–8501 Japan
†† Riken AIP　1–4–1 Nihon-bashi, Chuo-ku, Tokyo 103–0027
E-mail: †hirayama@ws.cs.kobe-u.ac.jp, ††sachio@carp.kobe-u.ac.jp, †††masa-n@cs.kobe-u.ac.jp

Abstract　In recent years, many applications which using face identification are developed. However, development load in creation of an application program using the face identification is high because of the tight coupling of each functions and lack of sharability of imaging device. To address this problem, in this research, we propose a new device named “face identification sensor box” which can realize loose coupling of function and high sharability. The proposed device uses cognitive API for face identification and notifies the result to application with Publish/Subscribe messaging model. Using this architecture, face identification results can be shared among with applications and design of application becomes easier.
Key words　Face identification, Cognitive API, Pub/Sub, IoT, Edge computing

1. はじめに

画像処理や AI 技術の急速な発展により，カメラなどにより取得した顔画像をもとに個人を識別する，コンピュータビジョンを用いた顔識別機能が様々なアプリケーションにおいて利用が進んでいる．顔識別とは，画像内に含まれる顔部分を特定・抽出し，抽出された顔に含まれる目・鼻といった個人を識別する際に特徴的な部分となるパーツの位置を特徴点として算出し，事前に登録された人物の顔が持つ特徴点と比較することにより，画像内に映っている人物を識別する技術である．顔識別は実際の利活用としてはユーザ認証に用いられる場合が多く，例として Apple 社のスマートフォンである iPhone X では顔識別を用いた本人認証によってスマートフォンのロックを解除や，オンライン決済を行う機能である Face ID が挙げられる [1]．さらに，テーマパークであるユニバーサル・スタジオ・ジャパンでは，年間パス保持者のパーク入園時に顔識別・顔認証により本人確認を行うことで，本人確認にかかる業務の効率化や不正利用の防止を行っている [2]．ユーザ認証以外にも，対話用ロボットに顔識別を搭載することで，対象ユーザを把握し，そのユーザに適した情報を提供する個人適用型の対話ロボットの実現や [3]，認知症患者などの無断外出・徘徊を防止する [4] などの様々な利用が進んでいる．

顔識別機能はアプリケーションへの新たな付加価値を与える一方，アプリケーションへ実装する際には，再利用性の低さに

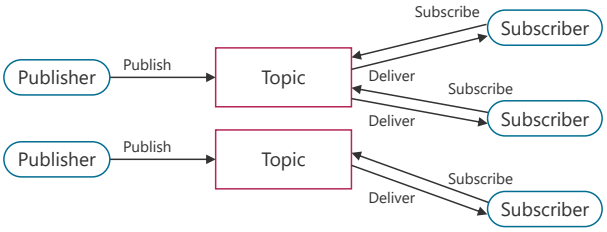

図 1　Pub/Sub の概念図

よる開発の負担や，カメラやデバイスといったリソース共有性の欠如などが問題となる．そこで本研究では，あらゆるアプリケーションから利用できるような顔識別用サービスを提供するセンサデバイスである，**顔識別センサボックス**を提案する．提案デバイスは，接続されたカメラで撮影を行いながら，顔が検出された際にコグニティブサービスを用いた顔識別を行い，映った人物を特定する．識別結果は，あらかじめサブスクライブされた自身を利用する複数のアプリケーションに対して，Pub/Sub 型のメッセージングモデルを用いて通知する．顔識別センサボックスを利用することで，同一空間で顔識別を必要とする複数のアプリケーションが，一つのセンサボックスを利用して，個々のタイミングで識別結果を利用することができる．また，顔識別に関する処理はセンサボックス側に委譲できるため，開発者はアプリケーションの設計にかかる手間を減らすことができ，提供するサービスの本質に注力が可能になる．また，アプリケーション内での依存を減らせるため，スケーラビリティの向上が期待できる．顔識別センサボックスの利用例として，例えば，オフィスにおいて，入口に 1 台顔識別センサボックスを設置し，カメラの前を人間が通過する際，顔識別を行うと同時に勤怠管理の登録や，部屋への入退室記録の作成を行い，対象ユーザの PC のロックが解除され，個人に合わせたスケジュールやリマインダーの実行などを行う，といった，一連のサービスが実現可能である．

本稿では，顔識別センサボックスの実現に向け，機能要求の設計およびプロトタイプ実装による提案手法の有効性を確認する．実際に作成したプロトタイプを用い，顔識別を利用する外部アプリケーションに対し顔識別サービスを提供できることを確認している．

2. 準　　備

2.1 Pub/Sub 型メッセージング

Pub/Sub（Publish/Subscribe）[5] とは，複数のマシンやサービス間が非同期にメッセージをやり取りするためのメッセージングモデルの一つである．図 1 に Pub/Sub の概念図を示す．Pub/Sub メッセージングモデルでは，メッセージを作成して送信する送信側のクライアントを *Publisher*，メッセージを受信する受信側のクライアントを *Subscriber* とよぶ．Publisher から送信するメッセージは，トピックという送信先に登録される．Subscriber は自身が受信したいトピックをあらかじめ購読（subscribe）する．トピックに登録されたメッセージは，そのトピックを購読している Subscriber に配信（Deliver）される．このメッセージングモデルにより，Publisher・Subscriber 双方が，互いの状態に依らないメッセージの送受信が実現できる．つまり，Publisher は Subscriber の状態にかかわらず逐次メッセージを送信すればよく，Subscriber は自身が必要なメッセージのみを受信することができる．これにより，アプリケーションのパフォーマンスやスケーラビリティの向上が実現する．

2.2 MQTT

MQTT（Message Queue Telemetry Transport）[6] は，2 者間の非同期通信を Pub/Sub 型メッセージングモデルで実現するプロトコルである．非同期であるため，ネットワークの信頼性に依らない通信が可能で，かつ軽量なプロトコルであるため，処理能力が高くない IoT（Internet of Things）デバイスなどで広く使われている．

MQTT では，ネットワーク内のエンティティのタイプとして，Broker と Client の 2 つが定義されている．Pub/Sub で定義されている Publisher と Subscriber が Client に相当し，Broker が Client の仲介役となりトピックごとにメッセージの宛先を管理する．実際の通信では，Publisher は Broker を通じて Subscriber にメッセージを送る．このとき，Publisher はそのメッセージの属するトピックを設定する．Broker はそのトピックをサブスクライブしている Subscriber に対してのみ，メッセージを送ることで Pub/Sub 型メッセージングを実現している．

2.3 コグニティブ API

クラウドコンピューティング技術の急速な発展に伴い，これまでエッジ側で実行していた様々な処理を，クラウドサービスでの代替が可能となった．このような中，近年爆発的に普及が進んでいる人工知能を活用する様々なサービスもクラウドサービスとして利用が可能となりつつある．人工知能を活用するクラウドサービスとして，視覚・音声・言語・知識などをもとに，そこに含まれている情報をコンピュータに認識させる，コグニティブサービスがある．コグニティブサービスでは従来コンピュータでの処理が困難であった様々な認知に関する機能を実現することができる．画像に基づくコグニティブサービスは，画像から様々な情報を抽出することが可能なサービスであり，多くのパブリッククラウドで提供されているサービスである．一般的にクラウドサービスは API という形で外部から利用することが可能であるため，これらのコグニティブ API についても利用者はこれらの API に画像を送信することで，画像内にある様々な情報を受け取ることができる．例えば，人間の顔画像から年齢や性別，顔表情による感情値を推測することができる．そして，家電や使用道具，食べ物，色等，様々な物品に対する画像認識もできる．また，画像背景から場所や天気に関するタグも出力することができる．

3. 提 案 手 法

本章では，**1.** で述べたような，様々なアプリケーションから利用できる顔識別デバイスの設計に関して具体的な説明を行う．

3.1 既存の顔識別アプリケーションの問題

顔識別機能はアプリケーションへの新たな付加価値を与える

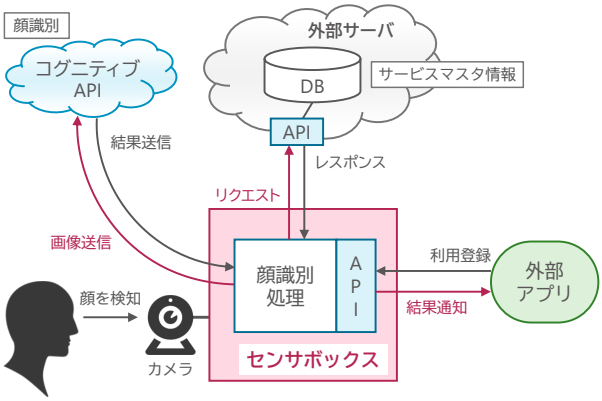

図 2　全体アーキテクチャの概略図

一方，アプリケーションへ実装する際には，開発者はサービス・撮影・顔識別機能をアプリケーションごとに実装する必要がある．そのため，各機能はアプリケーションと密結合し，結果として再利用性が低下してしまう問題がある．たとえば，本来は汎用的な機能である，撮影・顔識別の機能が再利用できない．また，一つのアプリケーションが一つのカメラを占有するため，同一空間内で複数の顔識別機能が必要となった場合，撮影対象は同一空間であるにも関わらず複数のカメラを必要としてしまうという問題もある．

3.2　顔識別センサボックス

既存アプリケーションの問題を踏まえ，本研究では，顔識別センサボックスの提案を行う．

顔識別センサボックスは，様々なアプリケーションがカメラや顔識別サービスを共有できるようにするための機能を提供するデバイスである． センサボックスは，同一の顔識別機能を複数のアプリケーションに提供できる必要がある．また，センサボックスが複数になった場合も考えると，各デバイスごとに顔識別に関する設定や，ユーザの追加，削除をまとめて行うことができることが求められる．そこで，センサボックスは，顔識別の処理を外部に移譲し，そちらで顔識別機能の実行や管理を行うことで，センサボックスごとの管理の負担を減らす．

センサボックスは，複数のアプリケーションから呼び出されることを想定して，同時に多くのポーリングが発生した場合でも対処できる必要がある．また，スケーラビリティを向上させるためには，各アプリケーションの状態に左右されずに，結果の通知をし続けられることが望ましいため，非同期に通信を行うことができる仕組みを採用する．

3.3　機 能 要 件

顔識別センサボックスを利用した顔識別は，以下の機能を持つ．

A.Pub/Sub によるサービス公開

B. 一定間隔での撮影と顔検出

C. コグニティブ API による顔識別

全体アーキテクチャの概略図を図 2 に示す．以下，各機能の説明を行う．

A. Pub/Sub によるサービス公開

アプリケーションは顔識別の機能を利用したい場合に，センサボックスに対して自身の利用登録（サブスクライブ）を行う．センサボックスは，登録されたアプリケーションを内部で記憶しておく．顔識別が完了し，その結果が得られたら，センサボックスはサブスクライブされたすべてのアプリケーションに対してその結果を通知する．その後，各アプリケーションは受け取った結果を用いて，自身の処理を行うことができる．センサボックスとアプリケーション間の通信には Pub/Sub を取り入れ，非同期で通信が行われるようにする．

また，サブスクライブする機能だけではなく，アプリケーションが顔識別の結果を必要としなくなった場合に備えて，センサボックスにはサブスクライブの解除を行う機能を設ける．

B. 一定間隔での撮影と顔検出

センサボックスは接続されたカメラから画像を取得し続ける．ただし，ここで得られたすべての画像に対して顔識別を行うことは非効率的であるため，顔識別を行う前段階として，センサボックス内で顔検出を行う．顔検出の処理では，**顔が存在するかどうか**のみ判定する．これにより，顔が映っている画像に対してのみ顔識別を実行することができる．

C. コグニティブ API による顔識別

B. の顔検出処理によって得られた画像をもとに，顔識別を実行する．顔識別にはコグニティブ API を利用し，センサボックスの外部に処理を移譲する．識別処理が終了したら，識別結果を受信する．

なお，コグニティブ API を利用して顔識別を行うためには，事前に利用するコグニティブ API に，識別対象ユーザの名前や顔写真といった情報を登録したり，識別に用いる機械学習モデルの訓練を実行しておくといった作業が必要となる．このような操作は，コグニティブ API のエンドポイントにリクエストを送信することで実行されるが，これらをアプリケーションが簡潔に行えるように，センサボックスにはラッパー API を設ける．また，コグニティブ API にユーザを登録する際，名前に日本語が利用できない，生年月日などの詳細なプロフィールを入力できないといった API 側の問題が発生することを想定し，外部サーバに独自のデータ形式を持つデータベースを構築しておき，コグニティブ API に必要な情報のみを送信して登録を行うと同時に，データベースのほうにも登録を行うようにする．さらに，データベースに格納された情報をアプリケーションが使えるようにするため，センサボックスを介してデータベースの参照を行う手段を API として用意する必要がある．外部サーバには，ユーザ情報のほかにも，コグニティブ API を利用するための API キーやエンドポイントなどといった各種設定情報も保持し，サービスマスタとして情報を提供し，各センサボックスからアクセスする際の利便性を上げる．

以上が，顔識別センサボックスに必要な機能である．

4.　プロトタイプ実装

4.1　実　　装

提案手法に基づき，顔識別センサボックスのプロトタイプを実装した．図 3 に全体の実装を示す．実装に用いた技術は以下の通りである．

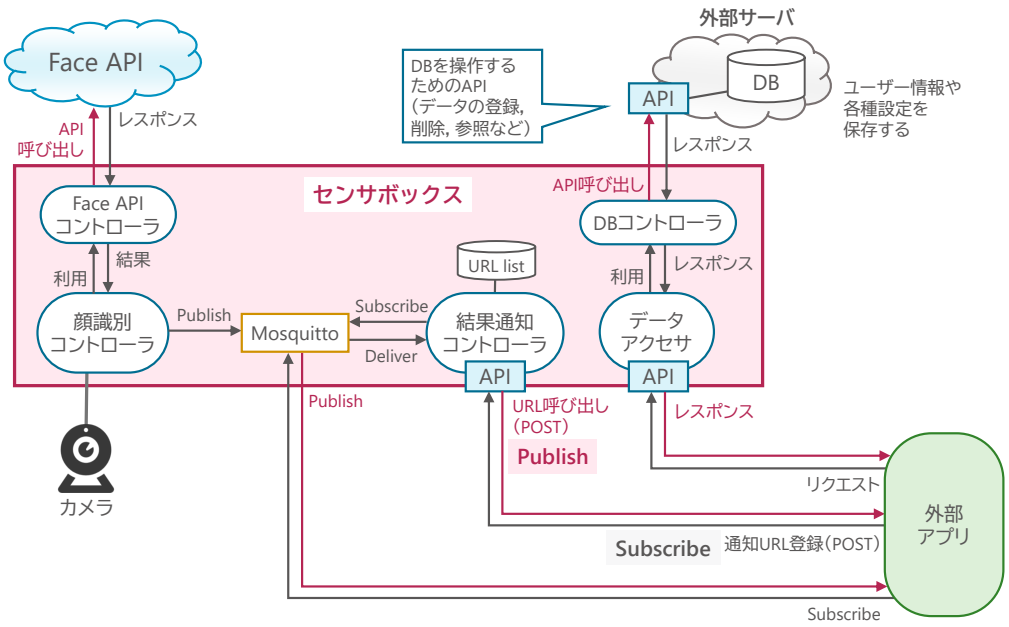

図 3　全体実装図

- 開発言語：Python 3.5，HTML5，JavaScript
- ライブラリ：Paho MQTT(注1)，Flask(注2)，OpenCV(注3)
- MQTT ブローカー：Mosquitto(注4)
- コグニティブ API：Microsoft Face API [7]

Paho MQTT，Mosquitto は，MQTT プロトコルのオープンソース実装である．Paho MQTT は Publish と Subscriber，Mosquitto は Broker の役割を果たす．Flask は，Python 用の Web アプリケーションフレームワークの一種であり，各機能の WebAPI 化に用いた．プロトタイプのエッジデバイスとして小型のシングルボードコンピュータの一つである Raspberry Pi 3(注5)（OS：Raspbian 9.4）を用いた．

4.2　識別結果の通知フロー

今回実装するプロトタイプでは，顔識別の結果を通知する部分が Pub 側，外部アプリケーションが Sub 側となり，双方で Pub/Sub 通信が行われる設計としている．顔識別コントローラは結果を Mosquitto に通知し，Mosquitto 内で特定のトピックに振り分ける．センサボックスの内部では，結果通知コントローラがそのトピックを Subscribe しており，非同期に顔識別の結果を受け取る．結果通知コントローラは，外部アプリケーションへの Publisher として機能し，あらかじめ Subscribe 済みの外部アプリケーションに対し，識別結果を任意の URL へ通知する．この処理により，結果通知コントローラが疑似的に Broker として働くことになる．

今回のプロトタイプではアプリケーションからの利用は想定していないが，顔識別センサボックスでは，MQTT による Pub/Sub にも対応しており，Mosquitto 自身が外部アプリケーションへの Publisher となり，外部アプリケーションに対し識別結果を通知することもできる．

4.3　動作準備

今回の実装ではコグニティブ API として Microsoft Azure のサービスの一つである Microsoft Face API（以下，Face API と略記）を用いる．Face API は顔の検出，顔の識別，表情分析などの機能を持ち，それらの機能をクラウド経由で手軽に利用することができる．Face API の顔識別機能を利用するには，まずグループ（＝ PersonGroup）を作り，そこに識別対象のユーザ（＝ Person）を追加していく必要がある．登録の際，各ユーザに Face API で用いられる ID（＝ personId）が割り振られる．登録された Person を指定し，顔画像をアップロードして，グループについて学習モデルの訓練を行うことで，顔識別が可能となる．これらの一連の操作は，指定の Face API エンドポイントに対して，各種パラメータを含めてリクエストを送ることで行われるが，これを簡単化するために，Face API に対する様々な操作をまとめたユーティリティ群（Face API コントローラ）を作成した．

また，外部のサーバに，ユーザ情報や各種設定値を格納するデータベースを構築し，API 経由でセンサボックスがアクセスできるようにする．データベースは，顔識別を利用するユーザの情報を［personId，ローカル ID，名前，誕生日，性別，ニックネーム，メモ］といった形で保持する．データベースの利用に関して，簡単化を目的とし，データ取得，書き込み，変更，削除，検索，ID 相互変換といった操作を行うためのユーティリティ群（DB コントローラ）を作成した．

さらに，これらのコントローラの機能を利用して，外部アプリケーションからユーザの登録や削除といった機能を使うための WebAPI（データアクセサ）を実装した．各コントローラを直接 API 化することを避けた理由は，データ登録の際，Face API への登録後，データベースにも登録する必要があるが，こ

(注1)：https://www.eclipse.org/paho/
(注2)：http://flask.pocoo.org/
(注3)：https://opencv.org/
(注4)：https://mosquitto.org/
(注5)：https://www.raspberrypi.org/

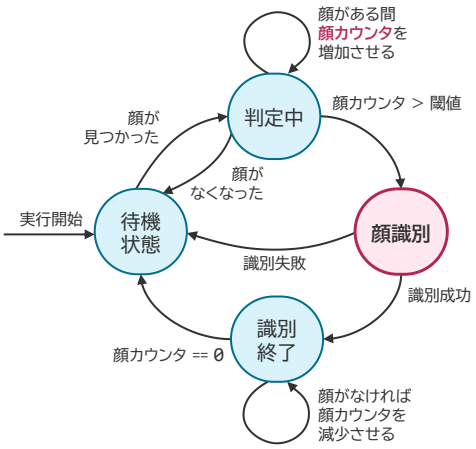

図 4　顔検出の動作

のような操作をひとまとめにし，必要な情報を POST で一括で登録可能にするためである．この結果，アプリケーション側の実装が簡素化できる．また，Face API 側の登録情報とデータベースの不一致を防ぐこともできる．データアクセサの機能として，ユーザ登録，削除だけでなく，personId とローカル Id の変換，各 Id に対応するユーザ情報の存在の確認や取得といった機能を持ち，必要に応じ外部のデータベースから情報を取得できる．

4.4　動 作 説 明

以下，**3.** における各機能要件と対応させ，プロトタイプの動作について説明する．なお，プロトタイプの Pub/Sub 通信には，URL 呼び出しによる方法を用いた．

A. Pub/Sub によるサービス公開

顔識別センサボックスを利用したい外部アプリケーションは，結果通知コントローラの API に対し，通知先 URL を登録する．この操作は `(DeviceURL):(PORT):/reg_app_url` といったエンドポイントに対して POST 通信をすることで実行する．登録後，顔識別が行われた際に，その結果はセンサボックスに登録されている全ての URL 宛てに通知される．また，各アプリケーションは，同様にして自身の登録をいつでも解除することができる．

B. 一定間隔での撮影と顔検出

センサボックスはカメラから画像を取得しつつ，顔検出を行う．顔検出には画像処理用ライブラリである OpenCV を用い，Haar Cascade [8] 分類器を作成し行う．分類器は予め用意されているものを利用した．取得画像内で顔検出が行われた場合，顔識別へ進む．

顔検出においては，単純に顔が発見された瞬間に顔識別を行うのではなく，安定した顔識別を目的とし，顔カウンタを導入する．顔カウンタを用いた顔検出の動作を図 4 に示す．顔カウンタは，画像内で顔検出が成功した場合に増加し，その値がある一定の閾値を超えたときにはじめてシステムとして顔が検出されたとみなす，また，一度顔が検出された後は，顔検出が成功しない間，顔カウンタを減少させ，その値が 0 になるまで次の顔の検出がシステムによって行われないようにする．これら一連の仕組みにより，ユーザがカメラの前で安定しているときの画像を取得できるため，顔識別が安定して行われる．他にも，後述する顔識別で用いる Face API は呼び出しの回数や間隔に制限が設けられているため，短時間に大量の顔識別を実行させられないが，顔カウンタを導入することで顔識別の回数を減らすことができるため，これに対処できる．

C. コグニティブ API による顔識別

顔識別コントローラは，顔検出の処理によって得られた画像を Face API コントローラに送信する．Face API 利用のための一連の処理は Face API コントローラにより行われ，最も識別の確信度（Confidence）が高い人物の personId が顔識別コントローラに返る．Face API 側が，登録されていない人物の顔画像を受け取った場合は，“unknown” という結果が通知されるようにした．

なお，Face API を利用して顔識別を行う前に，ユーザの登録を行う必要がある．ユーザの登録は，データアクセサ経由で行う．データアクセサは，ユーザ登録用に `(DeviceURL):(PORT):/register` といったエンドポイントをもち，そこに対して POST 通信で必要な情報を送信することにより，ユーザの新規登録を行う．また，顔画像の追加や削除に関しても同様にエンドポイントを持ち，外部からアクセスすることによって各種操作を行う．

顔識別の結果としてアプリケーションに通知されるのは personId のみで，名前などの情報は通知されない．そのため，ユーザの詳細な情報を利用したい場合は，データアクセサを利用し，personId をキーとし，情報を取得する必要がある．

4.5　サンプルプログラムおよび動作テスト

今回実装した顔識別センサボックスの機能を利用して，ユーザの登録用のアプリケーションと，顔識別の結果を受け取り，ユーザの名前を表示するアプリケーションの 2 つのクライアント用サンプル Web アプリケーションを作成した．これらのアプリケーションは顔識別センサボックスの外部から，各機能にアクセスする．実際にこれらのアプリケーションを使い，一連の動作テストを行った．

まず，ユーザ登録アプリケーションを用いて，ブラウザ上でユーザ登録を行う．アプリケーションは専用フォームに入力された各種情報をデータアクセサに送信する．データアクセサはデータベースにユーザ情報を登録するとともに，Face API への登録を行い，その結果，Face API からは personId が発行される．personId は

```
c0a094ab-e8b0-4368-...
```

のような 16 進数で表現されている．その後，ブラウザ上で顔画像の撮影を行い，同様にデータアクセサ経由で数枚の画像を Face API に送信する．

次に，結果取得アプリケーションを実行し，センサボックスの結果通知部分に自身の URL を送信する．登録が正常に完了すれば 200 の HTTP ステータスが返る．

ここまでの準備が完了したら，センサボックスのカメラに顔を向け，顔識別の処理を実行させる．顔識別の処理が終了する

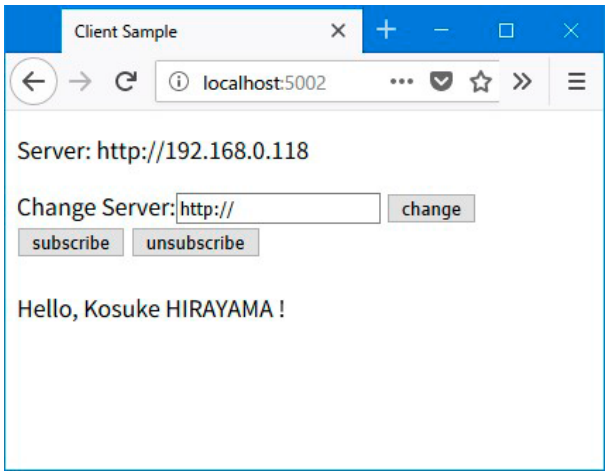

図 5　結果取得サンプルの画面

と，登録された URL に結果として personId が送信される．

結果取得アプリケーションは personId を受け取ると，それを名前に変換するために，データアクセサに送信する．データアクセサには personId をキーとして登録者の各種情報を JSON 形式で返す API があり，`/get_list_by_pId` という URL に対して POST 通信で personId を送信する．この結果，事前に登録した内容が含まれた下記 JSON が得られる．

```
{'note':None, 'personId':'c0a094ab-e8b0-...',
'gender':'m',
'localId':'hirakou', 'birthday':'1996/1/24',
'nickname':None, 'name':'Kosuke HIRAYAMA'}
```

最後に，アプリケーション側はそこから `name` をキーとして名前を抽出し，ブラウザ上に

```
Hello, Kosuke HIRAYAMA !
```

と表示する．スクリーンショットを図 5 に示す．

5. おわりに

本稿では，顔識別機能を持つアプリケーション作成において，開発効率，リソースの有効利用などを実現できる顔識別センサボックスを提案した．提案手法では，顔識別にクラウドサービスであるコグニティブ API を利用することで，高い顔識別能力の実現を簡単に利用でき，かつ管理を簡単化することができる．また，アプリケーションとの通信に Pub/Sub の仕組みを取り入れることで，識別結果を非同期に送信し，多くのアプリケーションからの利用に対応するスケーラビリティを確保できる．今後の課題としては，今回実装したシステムを，実際に規模の大きな外部アプリケーションに取り入れた際の性能評価や，顔識別機能だけでなく，様々なコグニティブサービスを実現可能なコグニティブセンサボックスへの拡張などが挙げられる．

謝辞　この研究の一部は，科学技術研究費（基盤研究 B 16H02908，18H03242, 18H03342，基盤研究 A 17H00731），および，立石科学技術振興財団の研究助成を受けて行われている．

文　献

[1] “iphone xs - face id - apple（日本）,” `https://www.apple.com/jp/iphone-xs/face-id/`. visited on 2018-10-12.
[2] “年間パスの顔認証システムについて | usj web チケットストア,” `https://www.usj.co.jp/ticket/apass/facecertification.html`. visited on 2018-10-12.
[3] “ペッパー君がさらに進化！顔認証で、特定人物への個別対応が可能に：markezine（マーケジン）,” `https://markezine.jp/article/detail/26819`. visited on 2018-10-15.
[4] “顔認証システムを使った医療・介護施設の無断外出防止策 | 認知症対策顔認証徘徊防止システム【lykaon】-リカオン株式会社-,” `https://www.facial-lykaon.com/topics/wandering_prevention.php`. visited on 2018-10-15.
[5] K. Birman and T. Joseph, “Exploiting virtual synchrony in distributed systems,” SIGOPS Oper. Syst. Rev., vol.21, no.5, pp.123–138, Nov. 1987.
[6] “Mqtt,” `http://mqtt.org/`. visited on 2018-10-12.
[7] “Face api - 顔認識ソフトウェア | microsoft azure,” `https://azure.microsoft.com/ja-jp/services/cognitive-services/face/`. visited on 2018-10-12.
[8] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol.1, pp.I–I, Dec. 2001.