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
SC2018-41（2019-03）

顔識別センサボックスとバーチャルエージェントを活用したスマート出退社サービスの開発

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　顔識別機能を取り入れたアプリケーションの利用の高まりを受け，我々は様々なアプリケーションから顔識別機能を利用するための IoT デバイスである「顔識別センサボックス」の開発を行っている．しかしながら，センサボックスを利用することでアプリケーション開発者が得られるメリットについては評価できていない．そこで，今回の研究では，顔識別センサボックスの有用性に関する検証を行う．また，それに伴い，実際にセンサボックスを利用するアプリケーションとして，業務管理システムである「顔パスワークログ」を開発する．
キーワード　顔識別，IoT，エッジコンピューティング，業務管理

Developing Smart Entry-Exit Service for Office Using Face Identification Sensor Box and Virtual Agent

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　To follow that many applications start to incorporate face identifying function, we have developed “Face identification sensor box” of an IoT device which provides face identification function to various applications. However, we have not yet considered what merits can application developers obtain. Therefore, in this research, we evaluate the usefulness of the sensor box. As a part of the evaluation, we develop a working management system “Kao-pass worklog” as an application which uses the sensor box practically.
Key words　Face identification, IoT, Edge computing, Working management

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．また，顔識別といった機能を外部からクラウドサービスの形式で利用できるコグニティブサービスも登場している．

このような背景のもと，我々は先行研究において，コグニティブサービスを活用した「顔識別センサボックス」の開発を行っている [1]．顔識別センサボックスは，コグニティブサービスを用いた顔識別機能を外部アプリケーションから手軽に利用するための IoT デバイスであり，アプリケーションに顔識別機能を実装する際の手間を減らし，スケーラビリティの向上を図ることを目的としている．

これまでの研究では，センサボックス自体の設計，開発を行い，それを活用するアプリケーションはこれからの課題となっている．そのため，実際にセンサボックスの利用によりアプリケーション開発にかかる負担を減らせるか，開発者にとってどのようなメリットがあるかといった点は評価できていない．

そこで，本研究では，顔識別センサボックスを活用した具体激なアプリケーションの開発を行うことで，有用性に関する検証を行う．具体的には，会社の勤怠管理システムに着目し，職場への出退勤を顔の識別のみで容易に記録できるサービスである**顔パスワークログ**を開発する．

顔パスワークログは，ユーザがセンサボックスの前を通過すると，顔識別を行って個人を識別し，出社の時刻を記録する．これをトリガとして，ヴァーチャルエージェント [2] [3] 及びオン

This article is a technical report without peer review, and its polished and/or extended version may be published elsewhere.
Copyright ©2019　by　IEICE

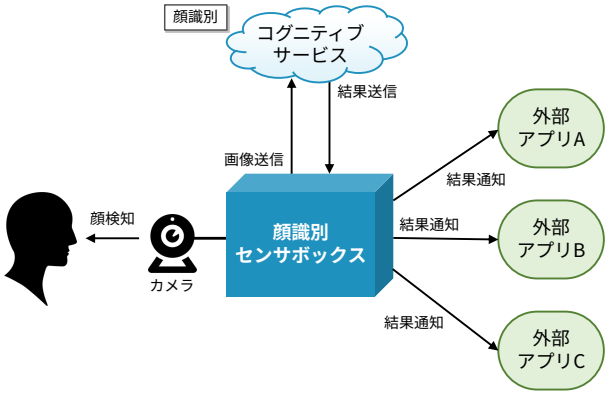

図 1　顔識別センサボックスの概念図

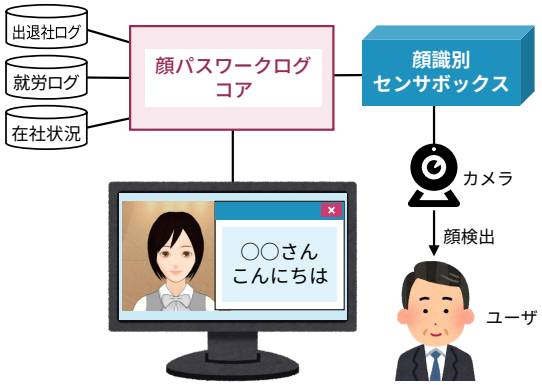

図 2　顔パスワークログの概念図

ラインスケジュールと連携し，出社時にはその日のスケジュールをユーザに語りかける．同様に，ユーザが退社する際にも，顔識別によって退社時刻を記録するとともに，ヴァーチャルエージェントがその日の就業の記録を会話によって聞き取る．ここで得られた就業内容も，出退社に加えて記録する．これにより，従来のタイムカードなどで煩雑であった就業管理の手間が大きく削減され，さらに，スケジュールの通知や作業の記録といった付加価値機能も提供できるようになる．

本稿では，顔パスワークログの設計及びプロトタイプ実装を行う．プロトタイプの実装に要したコードの行数と，センサボックス内部で動作するコード行数を比較することにより，センサボックスがどれだけ開発者の負担を軽減できるかを調べる．また，複数アプリケーションからのサブスクライブがあった際の動作検証を行うことで，スケーラビリティに関しての考察も行う．

今回の検証では，顔パスワークログを開発する際のコードの行数が，顔識別センサボックスを利用せずにフルスクラッチ開発した場合に比べて約 54% 削減された．また，顔識別センサボックスが複数のアプリケーションからのサブスクライブにも耐えうることを確認できた．これらのことから，顔識別センサボックスが，アプリケーション開発者の負担軽減およびスケーラビリティの向上に寄与できると考えられる．

2.　準　　備

2.1　顔識別センサボックス

顔識別センサボックスは，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ための IoT センサデバイスである．顔識別センサボックスの概念図を図 1 に示す．顔識別センサボックスは，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，あらかじめ自身をサブスクライブ（＝利用登録）している複数のアプリケーションに対して，Pub/Sub（Publish/Subscribe）[4] 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の目的は，アプリケーションの顔識別の処理をセンサボックスに委譲可能にすること，および，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共有し，個々のタイミングで識別結果を利用でき，顔識別という軸で連携することを可能にすることである．これにより，開発者のアプリケーション設計にかかる手間を削減し，さらに，アプリケーションと顔識別機能の間での依存を減らすことで，スケーラビリティの向上を目指している．

2.2　現在の業務管理システムの問題点

現代において，我が国の会社は，過労やサービス残業といった問題が多発した背景から，コンプライアンスが重要視されるようになっている．こうした中，従業員の業務状況を把握することは，働きすぎを防ぐためにも重要となっている．

業務時間の管理は，タイムレコーダを用いて，タイムカードに印字して行うことが一般的であった．しかし，タイムカードを用いた方法では，蓄積されたデータのコンピュータへの入力・管理が面倒である．また，他人のタイムカードを使った不正打刻も可能であるなど，効率が良いとは言えない．そこで，近年では，ICT を活用して出退社の時刻を打刻するシステムが登場してきている [5] [6]．このような勤怠管理システムを利用することで，データの管理が容易にできるほか，出退社の処理を IC カードや，生体認証など，様々な方法で行うことができる．

このような既存の勤怠管理システムの中には，顔識別による打刻が可能なものもある．しかしながら，そのためには専用の機器が必要である，顔識別機能がアプリケーションに密結合している，といった問題点を持っている．そのため，独自のアプリケーションと連携することによる機能拡張が難しいのが現状である．

これらの問題に対処し，顔識別によって出退社処理を行いつつ，かつ他の追加機能も容易に実装・連携できるようなシステムとして，本研究では，顔パスワークログを提案する．

3.　提案アプリケーション：顔パスワークログ

本章では，**1.** で述べた顔パスワークログの仕様・設計に関して具体的な説明を行う．

3.1　概　　要

本サービスは，**2.1** で述べた顔識別センサボックスにサブスクライブし，センサボックスによる顔識別の結果を得て，ユーザの出退社の記録などを行う．サービスの機能は，Web アプリケーションとして Web ブラウザ上から利用する．図 2 に顔パ

スワークログの概念図を示す．図 2 の**顔パスワークログコア**の部分が，ユーザにサービスを提供するプログラムを表している．下の画面は顔パスワークログのユーザーインタフェースを表す．顔パスワークログコアによって，画面上の顔パスワークログとヴァーチャルエージェントが動作する．ユーザはヴァーチャルエージェントとの会話を通じてシステムとの対話を行う．顔パスワークログは内部に出退社ログと日々の就労記録を蓄積し，また，現在の各ユーザの在社状況を保持している．ここで，顔パスワークログは顔識別センサボックスにサブスクライブを行っており，ユーザの顔が識別され，結果を受け取った際に，ユーザとの対話を開始する．その際，ユーザの在社状態に応じて，出社シナリオと退社シナリオのどちらかが実行される．

以下，それぞれのシナリオの流れを示す．

A. 出社シナリオ

（1） 出社してきたユーザに対し，「○○さん，こんにちは！」という挨拶を行う

（2） 挨拶を Web ブラウザ上に文字列として表示し，それと同時に，ヴァーチャルエージェントに発話させる

（3） 「こんにちは」と「私は○○ではありません」という選択肢のボタンを表示し，ユーザの回答を待つ

（4） ユーザは音声入力あるいはボタンを押すことにより回答をする

（5） ユーザが「こんにちは」を選択したら，そのユーザの出勤記録を日時とともに出退社ログのデータベースに記録する

（6） そのユーザの在社状況を「出社」に更新する

（7） 今日の予定をオンラインスケジューラから取得する

（8） 取得した予定情報をもとに，ユーザに「今日の予定は，13 時 30 分から，□□□，16 時 45 分から，△△△です」などと，今日の予定の通知を行う

（9） ユーザに「今日もお仕事頑張ってくださいね」と伝える

B. 退社シナリオ

（1） 退社しようとしているユーザに対し，「○○さん，お帰りですか？」という挨拶を行う

（2） 挨拶を Web ブラウザ上に文字列として表示し，それと同時に，ヴァーチャルエージェントに発話させる

（3） 「はい」，「いいえ」，「私は○○ではありません」という選択肢のボタンを表示し，ユーザの回答を待つ

（4） ユーザは音声入力あるいはボタンを押すことにより回答をする

（5） ユーザが「はい」を選択したら，「○○さん，お疲れさまでした」と伝える

（6） ユーザに「今日のお仕事はいかがでしたか？」と伝え，その日の就労ログを残すように促す

（7） ユーザは音声入力あるいはキーボードによる文字入力によって，就労ログを記入する

（8） 記入された就労ログをデータベースに保存する

（9） そのユーザの退勤記録を日時とともに出退社ログのデータベースに記録する

（10） そのユーザの在社状況を「退社」に更新する

（11） ユーザに「○○さん，さようなら」と挨拶する

3.2 各機能の説明

この章では，顔パスワークログが持つ各種機能を示す．

機能 1：入退社処理

顔パスワークログは，顔識別センサボックスから識別結果を受信し，誰が来たのかを判断する．出退社情報を保持するために，あらかじめ各ユーザの在社状況と出退社ログをそれぞれ記録するためのデータベースを用意し，ユーザの出退社時に，それらのデータベースの更新・追記を行い，出退社したユーザの名前とその時間を記録する．また，在社情報や出退社ログは，確認がしやすいように，ブラウザ上で表示できるようにする．

ここで記録された在社状況は，顔パスワークログの運用において，ユーザが来た際に，それが出社であるのか退社であるのかを判断するためにも使われる．

機能 2：ユーザとの対話

出退社処理の際などに，ブラウザ上にメッセージを表示し，ユーザとの対話を行う．また，メッセージの表示と同時に，その文章をヴァーチャルエージェントの音声合成機能を用いて読み上げる．対話は，語りかけと，問いかけの 2 つに分かれる．問いかけの場合は，ユーザに対して，その問いに対する回答を求める．この際，ブラウザ上で音声入力機能を起動し，声による回答ができるようにする．

機能 3：スケジュール通知

出社時，ユーザにその日の予定をアナウンスする．予定は，Google カレンダーといったオンラインスケジュールなど，外部に登録されたものを参照する．

機能 4：就労ログ

退社時，ユーザにその日の業務内容といった就労ログを残すよう尋ねる．ユーザは音声入力によって，就労ログを作成できる．作成されたログは，ユーザの名前や登録日時とともに，あらかじめ用意された就労ログ蓄積用のデータベースに記録される．また，就労ログも**機能 1** で述べた在社情報と同様に，ブラウザ上から確認できる．

機能 5：ログのエクスポート

機能 1，**機能 4** で述べた出退社ログと就労ログは，ブラウザ上で確認するだけではなく，CSV 形式のファイルとしてエクスポートすることができる．

4. プロトタイプ実装

4.1 実　　装

3.1，**3.2** で述べた仕様に基づき，顔パスワークログのプロトタイプを実装した．実装に用いた技術は以下のとおりである．

- 開発言語：Python 3.5，JavaScript
- ライブラリ：Flask(注1)，Flask-SocketIO(注2) SQLite3(注3)
- オンラインスケジュール：Google カレンダー
- API：Google Calendar API(注4)，Web Speech API(注5)

(注1)：http://flask.pocoo.org/

(注2)：https://flask-socketio.readthedocs.io/en/latest/

(注3)：https://www.sqlite.org/index.html

(注4)：https://developers.google.com/calendar/

(注5)：https://developer.mozilla.org/ja/docs/Web/API/Web_Speech_API

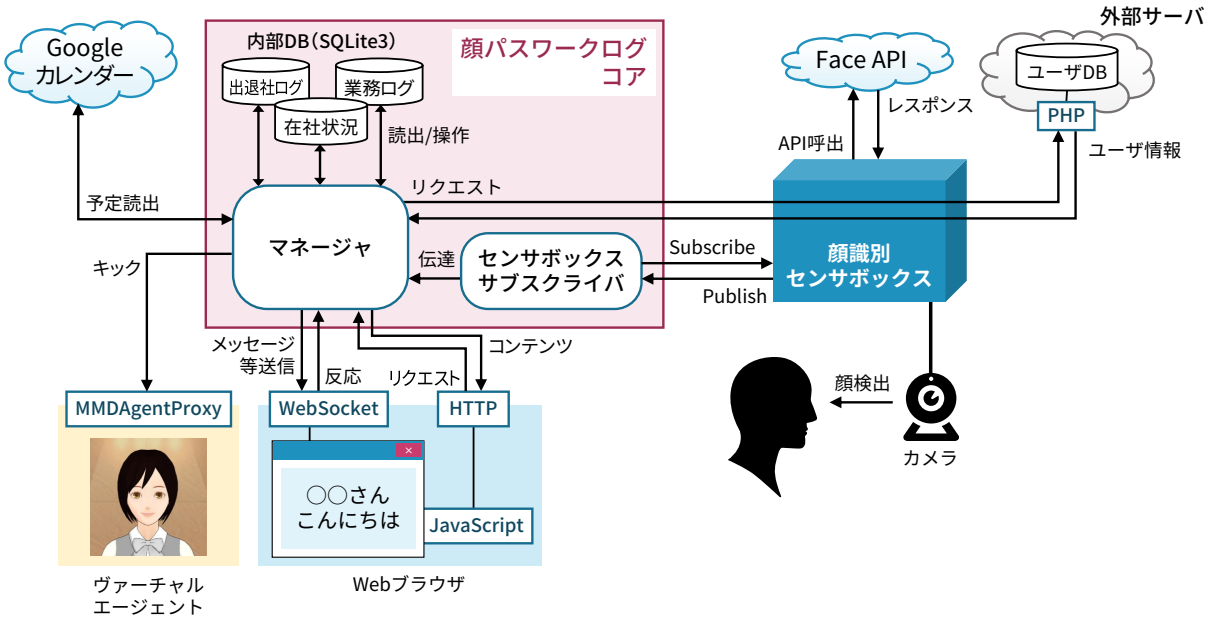

図 3　全体アーキテクチャ図

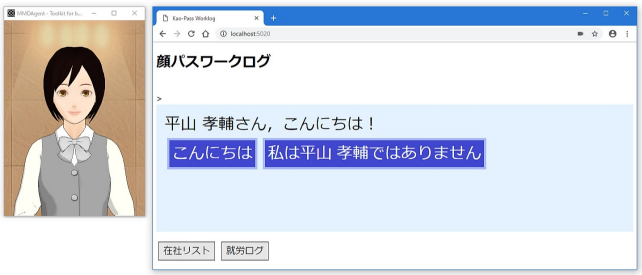

図 4　顔パスワークログのスクリーンショット

- Web ブラウザ：Google Chrome(注6)

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，顔パスワークログの Web アプリケーション化に用いた．Flask-SocketIO は，Flask アプリケーション上で Socket.IO を用いた WebSocket 通信を可能にするライブラリであり，ブラウザとの通信を行うために用いた．Google Calendar API は Google カレンダーの登録内容を取得・変更できる API である．Web Speech API はブラウザ上で音声入力を可能にする API であり，Google Chrome 上で動作する．

実装した顔パスワークログの全体実装図を図 3 に示す．また，スクリーンショットを図 4 に示す．図 3 について，「**顔パスワークログコア**」の部分が実際にユーザにサービスを提供するプログラムを表し，**マネージャ**は，Web ブラウザとの通信やヴァーチャルエージェントへの発話指示，内部データベースへのアクセス，Google カレンダーからの予定取得といった各種処理を行い，ユーザに顔パスワークログの各機能を提供する．また，**センサボックスサブスクライバ**は，顔識別センサボックスへのサブスクライブを行い，センサボックスから識別結果が送られた際に，マネージャに識別結果を伝達する．

顔パスワークログとともに動作するヴァーチャルエージェントとして，本プロトタイプでは MMDAgent(注7) を採用した．また，過去に研究室で開発された MMDAgentProxy というサービスを MMDAgent と同時に起動する．MMDAgentProxy によって，MMDAgent の各機能を呼び出す API が用意され，これら API にリクエストを送信することで，MMDAgent で表示されているエージェントに発話やポーズをさせることができる．

4.2　センサボックスの機能説明

顔識別センサボックスは，コグニティブ API として Face API [7] を利用し顔識別を行う．Face API では，事前にセンサボックスを通して，ユーザの情報や顔画像を登録する必要がある．また，Face API とは独立にユーザの詳細な情報を保持するためのユーザ DB を外部サーバに構築し，センサボックスはユーザ DB にアクセスすることにより，ユーザ情報を取得する．そのため，新規ユーザを登録する際は，Face API とユーザ DB の両方にユーザ情報を登録する必要があるが，センサボックスはユーザ登録用の API を持っており，API に外部から各種情報を POST 通信で送信することで，Face API およびユーザ DB への登録を同時に行うことが可能となっている．顔画像の登録やユーザの削除に関しても同様にセンサボックスの API 経由で行う．これらの API を利用することで，ユーザ登録用のアプリケーションを容易に作成することが可能であり，センサボックス内部にも簡易的なものが配備されている．

ユーザ DB は MySQL によって構築されており，PHP API を通して操作が行われる．テーブル構造は［uId，personId，name，reading（名前のヨミ），birthday，gender，nickname，description（メモ欄）］となっている．ここで，uId は顔識別センサボックス用の独自の Id，personId は Face API で使われる Id であり，センサボックスを利用するアプリケーションには主に uId を用いさせる．また，センサボックスは uId をキーとし

(注6)：https://www.google.com/intl/ja_ALL/chrome/

(注7)：http://www.mmdagent.jp/

て各種情報をユーザ DB から取得し，JSON 形式でアプリケーションに送信する API を持っている．センサボックスがアプリケーションに対して通知する顔識別結果は uId のみであり，その他の情報（名前など）が必要であれば，その API を利用して取得させる．

なお，以降では，識別対象となるユーザの登録は既に完了しているものとする．

以降，**3.** で述べた各機能の具体的な実装や動作について述べる．

4.3 動作準備

顔パスワークログを利用する前に，顔識別センサボックスを利用するための準備が必要である．顔パスワークログは図 3 のように，マネージャとセンサボックスサブスクライバの 2 つのコンポーネントに分かれており，センサボックスサブスクライバを用いて顔識別センサボックスへのサブスクライブを行うことで，利用準備が完了する．これ以降，主な処理や操作はクライアント用プログラムが行う．同時に，顔パスワークログとは別に，ヴァーチャルエージェントを起動させる．

顔パスワークログを使用するユーザは，まず顔識別センサボックスに接続されたカメラに顔を向ける．その後，ユーザの顔識別が行われ，センサボックスからは識別結果（uId）が Publish される．顔パスワークログはその結果を受け取ると，在社状況のデータベースを検索し，退社状態であるならば，そのユーザは出社してきたと判断し，出社状態であるならば，退社するところだと判断する．そして，顔パスワークログはそれをセンサボックスに送信することで，ユーザの名前を取得し，以降の出社シナリオ／退社シナリオの処理に用いる．

なお，出社シナリオでの予定通知は，Google カレンダーから Google Calendar API を用いて取得した予定を整形して作成する．

4.4 各種内部データベース

顔パスワークログは各ユーザの在社情報や，ユーザの出退社ログおよび就労ログを内部でデータベースに格納する．各データベースは SQLite3 によって構築されており，それぞれ Web ブラウザ上で表示・確認することができる．

在社情報データベースのテーブル構造は［uId，name，isattend（在社しているか）］となっており，ユーザの出退社処理と同時に更新される．なお，在社情報のデータベースには，センサボックスが用いる外部サーバ上のユーザデータベースと自動的に同期し，あらかじめ全てのユーザのエントリが作成される．

出退社ログデータベースは［uId，name，date，in/out（出社か退社か）］というテーブル構造を持っている．また，就労ログデータベースのテーブル構造は［uId，name，date, log（ログの内容）］となっている．両者とも，date（日時）はログの追加と同時に，自動的に付加される．

また，出退勤ログおよび就労ログは CSV 形式でエクスポートが可能であり，Web ブラウザ経由でダウンロードすることができる．

表 1　コード行数

	センサボックス	顔パスワークログ
Python	714	511
HTML/JavaScript	275	496
PHP	281	0
合計	1,270	1,007

5. 評　　価

5.1 センサボックスの機能要求

顔識別センサボックスの目的は，アプリケーションに非依存な顔識別機能をサービスとして切り出し，様々なアプリケーションから利用・共有できるようにすることである．これによって，アプリケーション開発者の開発コストを下げるとともに，システムのスケーラビリティを向上させることを狙っている．本章では，センサボックスの活用によりどれだけの開発コストが削減できるか，また，スケーラビリティの向上が見込めるかについて，実験的な評価を行う．

開発コストの削減評価については，4. で開発した顔パスワークログを，センサボックスを使う場合と使わない場合で，アプリケーションのコード行数がどれだけ削減されるかを計測し，考察を行う．また，スケーラビリティについては，多くのアプリケーションからサブスクライブされるシチュエーションを想定したテストを行い，これを考察する．

5.2 コード行数の検証

顔パスワークログの開発コストが，顔識別センサボックスの利用により削減できるかを検証するために，プログラムのコード行数を測定する．

表 1 に顔識別センサボックスおよび顔パスワークログの動作に必要なコードの行数を示す．これらの数値は，空行やコメントを除いた純粋なコードの行数である．また，プログラムの実行において直接関係のないような部分（例えばデバッグ用の print 文など）は事前にコメントアウトしたうえで計測している．

ここで，センサボックスの動作に必要なコードのうち，顔パスワークログの動作に必要な，顔検出，顔識別，サブスクライブ，ユーザ情報の読み取りといった機能に関係するコードのみを抽出すると，その行数は Python，HTML/JavaScript，PHP を合わせて 475 となった．また，ユーザの登録作業を行う際に必要なコードの行数を同様に測定すると 691 であった．この 2 つを合わせると 1166 となる．

この結果から計算すると，今回の顔パスワークログの開発においては，事前のユーザ登録についても考えた場合，センサボックスを利用せずにフルスクラッチ開発した場合と比較して，コード行数が $\frac{1166}{1166+1007} = \frac{1166}{2173} = 0.536$ より，約 54% 削減できたことになる．

5.3 複数のアプリケーションからのサブスクライブ検証

スケーラビリティに関する実験的な評価を行うため，顔識別センサボックスとサブスクライブするプログラムが，今までの章の場合のように 1 対 1 ではなく，1 対多の場合でもサブスク

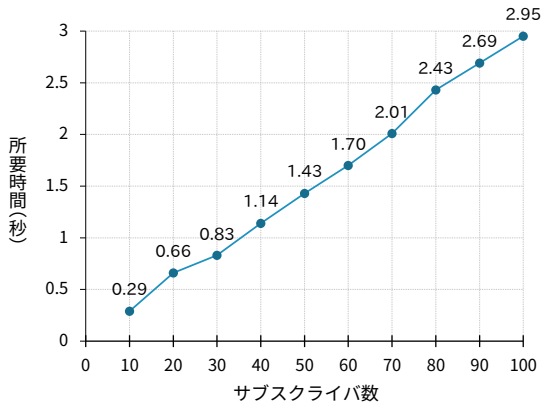

図 5　サブスクライバ数と処理時間の関係を示すグラフ

ライブ処理及び複数アプリケーションへの識別結果の通知を正しく行うことができるかを検証する．

検証のために，センサボックスにサブスクライブし，結果を受信したら終了する，というシンプルな Web アプリケーションを作成し，同一のコンピュータ上で，自身のポート番号を変更させながら多重起動させる．具体的には，`http://192.168.0.xxx:50000`，`http://192.168.0.xxx:50001`，`http://192.168.0.xxx:50002`，といったように 1 つずつ異なるポートが割り当てられたアプリケーションを同時に起動させ，その後，センサボックスからの識別結果のパブリッシュによりこれら全てのアプリケーションが同じタイミングで正常終了するかを確かめることによって，識別結果の通知が正しく行われたかを確かめる．

今回の検証では，サブスクライブするアプリケーションの数（サブスクライバ数）を 10 から 100 まで 10 刻みで増やしながら，識別結果のパブリッシュが正常にできるかの確認を行いつつ，全アプリが結果を受信するのに要した時間 T を測定する．具体的には，N 個のアプリを同時起動させ，1 個目のアプリが結果を受信した時刻を t_1，N 個目のアプリが受信した時刻を t_2 としたときの $T = t_2 - t_1$ を調べる．図 5 に検証結果のグラフを示す．縦軸が T，横軸がサブスクライバ数を表す．

5.4 考　　察

コード行数の検証により，今回の顔パスワークログの開発コストは，顔識別センサボックスの利用により削減されたといえる．削減できた要因は，顔識別やユーザ登録といった，顔識別を利用するアプリケーションにとって必要不可欠な機能の実装を省くことができたためであるといえる．これを踏まえると，顔パスワークログに限らず，センサボックスの利用により，様々なアプリケーションの開発において顔識別部分の実装にかかる開発コストが軽減できると考えられる．

また，複数のアプリケーションからのサブスクライブの検証により，顔識別センサボックスは，サブスクライバ数が多くなっても，それぞれに正しく識別結果を送信できることが確認できた．また，サブスクライバへ情報を伝達するのに要する時間の増加がほぼ線形であることから，アプリケーションの数が増加してもセンサボックスのパフォーマンスが低下しないといえる．このことから，センサボックスは複数のアプリケーションとの円滑な連携が可能であり，結果として，スケーラビリティの向上という目的を果たすことができると考えられる．

6. おわりに

本稿では，先行研究である顔識別センサボックスを利用して，顔パスワークログという業務管理システムの試作を行った．顔パスワークログを利用することで，顔識別による出退社管理やその他のサービスを提供できる．また，2 つの検証を通じて，センサボックスが，自身の持つべき機能要求を満たせているかを確認した．今回行った検証により，センサボックスは，アプリケーション開発者のコストの削減，およびスケーラビリティの向上という 2 つの機能要求を満たせるものと期待できる．今後も，センサボックスが顔識別を利用するシーンにおいて有益であるかを検証するために，より詳細なテストや，規模の大きいアプリケーションの試作を行いたいと考える．

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] 平山孝輔，佐伯幸郎，中村匡秀，“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，” 電子情報通信学会技術報告書，no.SC2018-23，pp.1–6，Nov. 2018．神戸大学 瀧川記念学術交流会館．

[2] S. Tokunaga, K. Tamamizu, S. Saiki, M. Nakamura, and K. Yasuda, “VirtualCareGiver: Personalized smart elderly care,” International Journal of Software Innovation (IJSI), vol.5, no.1, pp.30–43, Oct. 2016. DOI: 10.4018/IJSI.2017010103, http://www.igi-global.com/journals/ abstract-announcement/158780.

[3] H. Horiuchi, S. Saiki, S. Matsumoto, and M. Nakamura, “Virtual agent as a user interface for home network system,” International Journal of Software Innovation, vol.3, no.2, pp.24–34, April 2015.

[4] K. Birman and T. Joseph, “Exploiting virtual synchrony in distributed systems,” SIGOPS Oper. Syst. Rev., vol.21, no.5, pp.123–138, Nov. 1987.

[5] “No.1 勤怠管理・シフト管理システム「ジョブカン」,” `https://jobcan.ne.jp/`. visited on 2019-02-08.

[6] “勤怠管理システム「king of time(キングオブタイム)」簡単操作のクラウド勤怠管理で働き方が変わる。,” `https://www.kingtime.jp/`. visited on 2019-02-13.

[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.