平成27年度 卒業論文

琉球大学工学部情報工学科

125719G 長倉貴洋

指導教員 谷口祐治

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　研究背景と目的

スマートデバイスは, 2010 年以降から急速な発展と普及を遂げている. 日本国内における, 近年の情報通信機器の世帯保有状況 [1] は, 2010 年におけるスマートフォンの保有率が 9.7%, タブレット型端末が 7.2%であるのに対し, 2015 年ではスマートフォンが 62.6%, タブレット型端末が 21.9%となっており, 過去 5 年間でスマートフォンでは 6.45 倍, タブレット型端末では 3.04 倍の爆発的普及を実現している.(図 1.1 参照.) 近年は, 一人でスマートデバイスを複数台所有していることも珍しくない.

また, スマートフォン, タブレット型端末は, それらが登場する以前に普及していたフィーチャーフォンや PHS とは圧倒的な機能差があり, 非常に多機能なデバイスである. その多機能性を利用したシステムも数多く開発されており, GPS を利用したナビゲーションシステムや複数端末との通信を利用したソーシャルゲームなど, スマートデバイス普及以前では実現が難しかった様々なシステムが実用化され, 多くのユーザに利用されている.

本研究では, スマートフォン・タブレット型端末のポータビリティの高さとそれらの多機能性に着目し, 「ポータブルデバイスを入力とする情報収集アプリケーションの開発, および収集した情報を解析する Web アプリケーションの開発」を行い, スマートデバイスが収集する情報の正確性, データ解析の速度を調査する. また, それらを連携させたシステムを試験し, 集積・解析ツールとして利用可能であるかを判定することを目的とする.

ポータビリティに特化しているスマートデバイスが収集し得る情報郡を集積し, それらを解析することによって, 「人々の動きの流れ」や「特定の地域で活動する時間帯」, 「どのような事物に興味を示すのか」など, 多種類の有用な市場情報を獲得することが可能となる. 獲得した情報を適切に利用することで, 市場における現状況の改善や新しい需要の創出等による経済的発展が期待できる.

スマートデバイスで収集可能な多数の情報の中から, 今回は“ デバイスで撮影した写真データに記されている文字列 ”を対象とした情報収集, および収集データの解析を行う. “ 画像データ内の文字列 ”を収集対象としたのは, 「沖縄県が全国有数の観光地である」こと, 「観光客が有名観光地の名称が入った場所で撮影を行う」こと, 「観光客がどの時間帯にどの観光地へ訪れるのかを解析したデータは, 県経済における観光収入の比率が大きい沖縄県において非常に有用な情報となる」ことが主な理由である.

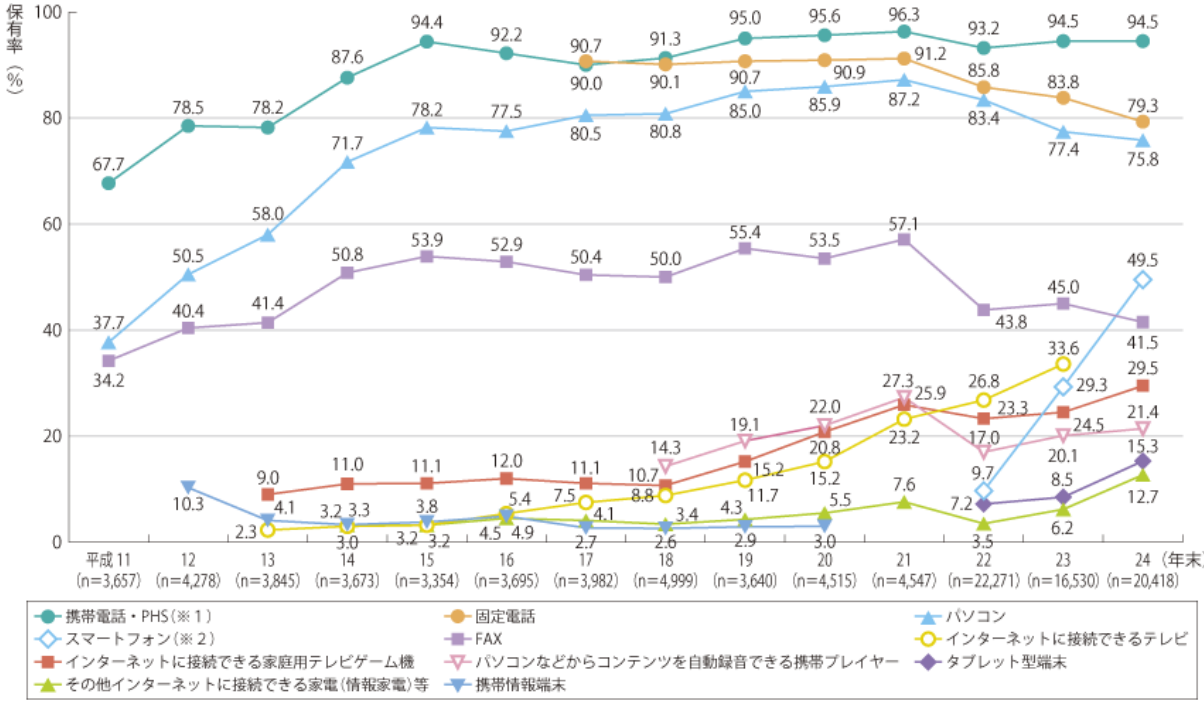

図 1.1: 情報通信端末の世帯保有数の推移 (出典：総務省「平成 25 年度版 情報通信白書のポイント」)

1.2 スマートデバイス

2010 年以降から普及しているスマートデバイスであるが, 現在, それらに搭載されている代表的なオペレーティングシステムとして, iOS, Android が挙げられる. さらに, タブレット型パーソナルコンピュータの開発, 販売競争も激化しており, 容易に持ち運びが可能なパソコンとして Microsoft Windows 10 が普及し始めている.

表 1.1 に, 2013 年から 2015 年にかけての, スマートデバイス用 OS の市場占有率 [2] のうち, iOS/Android/Windows の推移を示す.

1.2.1 iOS

iOS とは, Apple Inc. が開発およびリリースしている, iPhone/iPod touch/iPad/iPad mini/Apple TV のオペレーティングシステムである.

2007 年 6 月に発売された初代 iPhone と同時に提供され, 現在でも開発, 提供されている.(表 1.2 参照.)

iOS の最大の特徴は, Max OS X と連携が取れる点である.s iPhone で撮影した画像や動画を Mac に送信し編集する, iPhone にかかってきた電話を Mac で受ける, Handoff で Mac で編集していた書類を iOS デバイスで編集するなど, iOS と Mac OS X の高い連携度によって, デバイスに依らない、柔軟性に富んだ作業を実施することが可能となっている.

年月	iOS(%)	Android(%)	Windows(%)
2013 年 1-3 月	58.90	25.00	1.29
2013 年 4-6 月	58.70	25.05	1.22
2013 年 7-9 月	55.53	27.68	1.01
2013 年 10-12 月	54.91	33.39	0.57
2014 年 1-3 月	53.54	35.84	0.57
2014 年 4-6 月	48.32	41.06	1.65
2014 年 7-9 月	45.48	44.14	2.53
2014 年 10-12 月	43.99	46.03	2.27
2015 年 1-3 月	42.37	47.30	2.49
2015 年 4-6 月	39.49	51.74	2.26
2015 年 7-9 月	40.22	52.34	2.53
2015 年 10-12 月	36.86	55.68	2.87

表 1.1: スマートデバイスのオペレーティングシステムの市場占有率の推移

1.2.2 Android

Android は, Google が開発およびリリースしている, Linux カーネルをベースとした, スマートデバイス用オペレーティングシステムである.

2007 年 11 月に Android 1.0 が提供され, 現在でも開発, 提供されている.(表 1.3 参照.)

Android は, Android SDK に対応しているスマートデバイスならば, どれでもその OS として無償で利用することが可能である. また, 他のスマートデバイス向けオペレーティングシステムよりも, ミドルウェアやアプリケーションを作成する際の環境を構築することが容易である. その高い自由性と対応性により, 2016 年現在で最も利用されているスマートデバイス用オペレーティングシステムである.

1.2.3 Microsoft Windows 10

Microsoft Windows 10 とは, Microsoft Corporation が開発およびリリースしている, Windows シリーズに属するオペレーティングシステムである.

1985 年 11 月に Windows 1.0 が発表され, 現在でも開発, 提供されている.(表 1.4 参照.)

Windows シリーズは, デスクトップ用オペレーティングシステムとしては, 2015 年 7 月時点で約 90%のシェアを誇り, 現在最も利用されているデスクトップ用オペレーティングシステムである.

Windows 8 以降は, タブレット型スマートデバイスでも利用することが可能となっており, 持ち運びの便利なタブレット型デバイスに Windows OS を搭載し利用するパソコンユーザも少なくない.

1.2.4 本研究で使用するスマートデバイス用オペレーティングシステム

本研究では, 使用するスマートデバイス用オペレーティングシステムを iOS とする.
その理由として, 以下を挙げる.

- ポータビリティの高さ

  iOS デバイスは, スマートデバイスでは現在 2 位の市場占有率を誇り, そのポータビリティはデータの回収に高い信頼性があると判断した.

- 開発環境の構築の容易さ

  琉球大学工学部情報工学科では, 講義に Macbook を使用しており, iOS/OS X 用アプリケーションの開発ツールスイーツである Xcode と Objective-C のリファレンスなど, iOS デバイスのアプリケーションを作成するための環境を構築することが容易であるため.

  また, 講義で iOS アプリケーションの開発言語である Objective-C を学んだことも, iOS デバイスを使用する要因の一つである.

- Android

1.3 論文の構成

本論文では, 第二章「技術概要」にて本研究にて使用したプログラミング言語やライブラリを解説する.

第三章「実験」では, どのようにして iOS アプリケーションと Web アプリケーションが連携しているのかを示す.

第四章「本研究の利用例, 利用アイデア」では, 本研究をどのように活かせるのかを考察する.

バージョン	リリース年月	対応デバイス
iPhone OS 1.0	2007 年 6 月	iPhone
iPhone OS 1.1	2007 年 9 月	iPhone, iPod touch
iPhone OS 2.0	2008 年 7 月	iPhone/3G, iPod touch
iPhone OS 2.1	2008 年 9 月	iPhone/3G, iPod touch
iPhone OS 2.2	2008 年 11 月	iPhone/3G, iPod touch
iPhone OS 3.0	2009 年 6 月	iPhone/3G/3GS, iPod touch
iPhone OS 3.1	2009 年 9 月	iPhone 3G/3GS, iPod touch
iPhone OS 3.2	2010 年 4 月	iPad
iOS 4.0	2010 年 6 月	iPhone 3GS/4, iPod touch, Apple TV
iOS 4.1	2010 年 9 月	iPhone 3GS/4, iPod touch, Apple TV
iOS 4.2.1	2010 年 11 月	iPhone 3GS/4, iPod touch, iPad/iPad 2, Apple TV
iOS 4.3	2011 年 3 月	iPhone 3GS/4, iPod touch, iPad/iPad 2, Apple TV
iOS 5.0	2011 年 10 月	iPhone 3GS/4/4S, iPod touch, iPad 2, Apple TV
iOS 5.1	2012 年 3 月	iPhone 3GS/4/4S, iPod touch, iPad/iPad 2, Apple TV
iOS 6.0	2012 年 9 月	iPhone 4/4S/5, iPod touch, iPad/Mini, Apple TV
iOS 6.1	2013 年 1 月	iPhone 4/4S/5, iPod touch, iPad/Mini, Apple TV
iOS 7.0	2013 年 9 月	iPhone 4S/5/5C/5S, iPod touch, iPad/Air/Mini/Mini 2, Apple TV
iOS 7.1	2014 年 3 月	iPhone 4S/5/5C/5S, iPod touch, iPad/Air/Mini/Mini 2, Apple TV
iOS 8.0	2014 年 9 月	iPhone 5/5C/5S/6/6 Plus, iPad/Air/Mini/Mini 2, Apple TV
iOS 8.1	2014 年 10 月	iPhone 5/5C/5S/6/6 Plus, iPad/Air 2/Mini/Mini 2/Mini 3, Apple TV
iOS 8.2	2015 年 3 月	iPhone 5/5C/5S/6/6 Plus, iPad/Air 2/Mini/Mini 2/Mini 3, Apple TV
iOS 8.3	2015 年 4 月	iPhone 5/5C/5S/6/6 Plus, iPad/Air 2/Mini/Mini 2/Mini 3, Apple TV
iOS 8.4	2015 年 6 月	iPhone 5/5C/5S/6/6 Plus, iPod touch, iPad/Air 2/Mini/Mini 2/Mini 3, Apple TV
iOS 9.0	2015 年 9 月	iPhone 5/5C/5S/6/6 Plus/6S/6S Plus, iPod touch, iPad/Air 2/Mini 2/Mini 3/Mini 4, Apple TV
iOS 9.1	2015 年 10 月	iPhone 5/5C/5S/6/6 Plus/6S/6S Plus, iPod touch, iPad/Air 2/Mini 2/Mini 3/Mini 4/Pro, Apple TV
iOS 9.2	2015 年 12 月	iPhone 5/5C/5S/6/6 Plus/6S/6S Plus, iPod touch, iPad/Air 2/Mini 2/Mini 3/Mini 4/Pro, Apple TV

表 1.2: iOS のメジャーバージョンと対応デバイスの変移

バージョン	リリース年月
Android 1.0	2008 年 9 月
Android 1.1	2009 年 2 月
Android 1.5 Cupcake	2009 年 4 月
Android 1.6 Donut	2009 年 9 月
Android 2.0 Eclair	2009 年 10 月
Android 2.1 Eclair	2010 年 1 月
Android 2.2 Froyo	2010 年 5 月
Android 2.3 Gingerbread	2010 年 12 月
Android 3.0 Honeycomb	2011 年 2 月
Android 3.1 Honeycomb	2011 年 5 月
Android 3.2 Honeycomb	2011 年 7 月
Android 4.0 Ice Cream Sandwich	2011 年 10 月
Android 4.1 Jelly Bean	2012 年 7 月
Android 4.2 Jelly Bean	2012 年 11 月
Android 4.3 Jelly Bean	2013 年 6 月
Android 4.4 KitKat	2013 年 10 月
Android 5.0 Lollipop	2014 年 11 月
Android 5.1 Lollipop	2015 年 3 月
Android 6.0 Marchmallow	2015 年 10 月

表 1.3: Android のメジャーバージョンとリリース年月の一覧

リリースネーム	バージョン	リリース年月	特徴
Windows 1.0	1.0	1985 年 11 月	MS-DOS コマンドを入力するのではなく, 画面上をマウスでクリックして動作させる
Windows 2.0	2.0	1987 年 12 月	コントロールパネル, キーボードショートカットの導入
Windows 3.0	3.0	1990 年 5 月	プログラムマネージャ, ファイルマネージャ, の導入
Windows 95	4.00	1995 年 8 月	スタートメニュー, タスクバー, ウィンドウの最小化/最大化/閉じるボタンの導入
Windows 98	4.10	1998 年 6 月	コンシューマ向けに設計された初めてのバージョン
Windows 2000	NT 5.0	2000 年 2 月	ファイル暗号化システム, Windows File Protection の導入
Windows ME	4.90	2000 年 9 月	システム復元機能の導入
Windows XP	NT 5.1	2001 年 10 月	二列で構成されたスタートメニュー, タスクバーのタスク結合機能の導入
Windows Vista	NT 6.0	2007 年 1 月	透明なウィンドウなどの視覚効果の導入
Windows 7	NT 6.1	2009 年 10 月	サムネイルプレビュー, 最近開いたファイルをリスト表示するジャンプリストの導入
Windows 8	NT 6.2	2012 年 10 月	タッチスクリーンデバイスへの対応, スタートボタンを廃止し, スタートスクリーンを導入
Windows 8.1	NT 6.3	2013 年 10 月	スタートボタンの復活, 最大 4 つのアプリケーションを並べて使用できる機能の導入
Windows 10	NT 10.0	2015 年 7 月	生体認証テクノロジーの導入

表 1.4: Microsoft Windows のメジャーバージョンとリリース年月の一覧

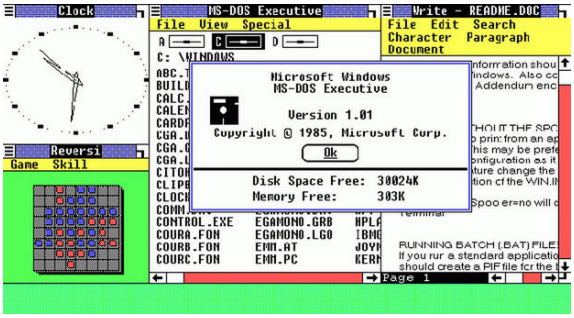

図 1.2: Windows 1.0 のデスクトップ画面

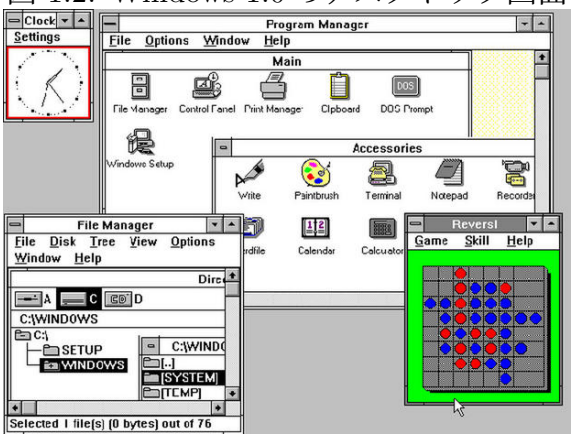

図 1.3: Windows 3.0 のデスクトップ画面

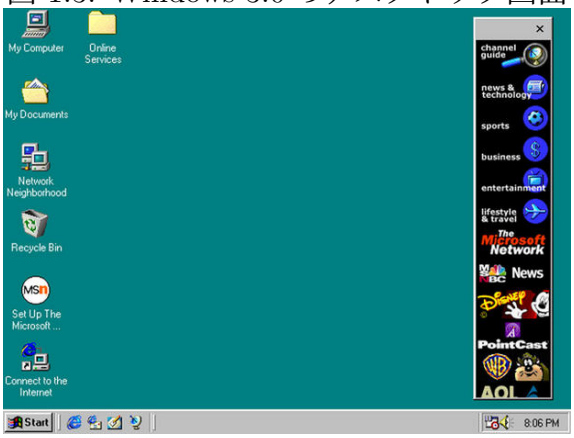

図 1.4: Windows 98 のデスクトップ画面

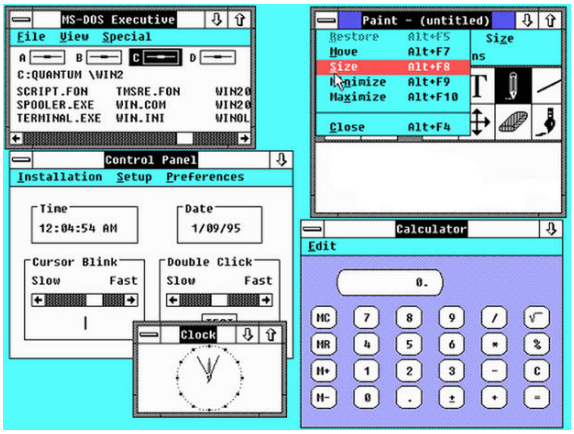

図 1.5: Windows 2.0 のデスクトップ画面

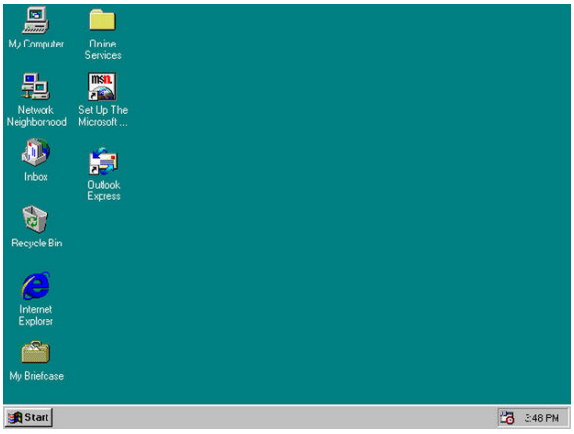

図 1.6: Windows 95 のデスクトップ画面

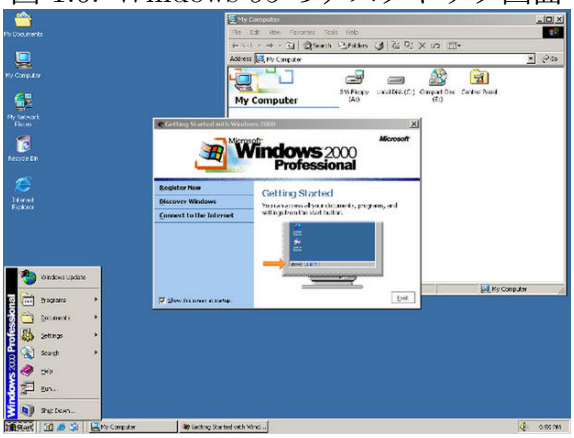

図 1.7: Windows 2000 のデスクトップ画面

図 1.8: Windows XP のデスクトップ画面

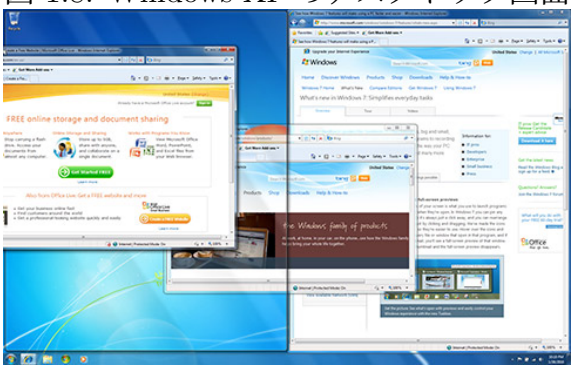

図 1.9: Windows 7 のデスクトップ画面

図 1.10: Windows 8.1 のデスクトップ画面

図 1.11: Windows Vista のデスクトップ画面

図 1.12: Windows 8 のデスクトップ画面

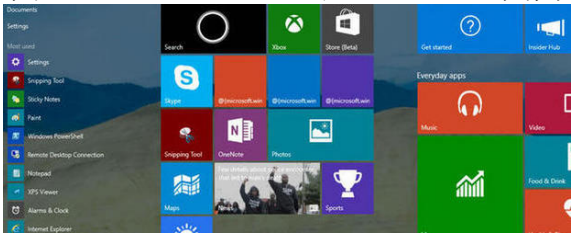

図 1.13: Windows 10 のデスクトップ画面

第2章　技術概要

2.1　iOSアプリケーション

2.1.1　Objective-C

Objective-C[1] とは, C言語をベースとして開発されたプログラミング言語である. NEXTSTEP[1] と Mac OS X[2] に標準付属されており, 現在は主に Mac OS X と iOS 上で動作するアプリケーションを開発する際に利用されている. Objective-C は GCC によってコンパイル可能であるため, 基本的には UNIX, Linix 系 OS や, GCC コンパイラを使用できる環境であればプログラムを動作させることが可能である.

“ C言語に Smalltalk をマクロ的拡張として施した言語 ”であり, “ C言語により近い拡張を施された C++と異なり, C言語とオブジェクト指向が混在した言語 ”であると述べるほうが適切である.

if/for/while などの制御文や, int/char/float などのスカラー型, 関数記法, 宣言や代入といった基本的な文法は C言語に準拠しているが, “ オブジェクト指向は Smalltalk の概念を借用したもの ”であり, その大きな特徴である“ メッセージパッシングによるメソッド呼び出しを利用した, 記述力に優れたオブジェクトシステム ”を継承している.

(A) Objective-C の特徴

Objective-C によるオブジェクト指向開発環境 [2] は, 以下の 3 要素から成り立つ.

- オブジェクト

  オブジェクト指向言語を利用したプログラムは, オブジェクトと呼ばれる要素を中心に構成される. オブジェクトとは, “ データと, そのデータを利用したりデータに作用する特定の関数による操作を関連付けたもの ”をいい, これらの操作はオブジェクトのメソッドと呼ばれる. メソッドが作用するデータをインスタンス変数という. よって, オブジェクトは“ インスタンス変数とメソッドを, 自己完結型のプログラミング単位にまとめたもの ”と言い換えることができる.

  Objective-C では, “ オブジェクトのインスタンス変数はオブジェクト内に存在し, 一般には, オブジェクトのメソッドによってのみ作用される ”. 他のメソッドがイン

[1] NeXT コンピュータに搭載されたオブジェクト指向マルチタスクオペレーションシステム.
[2] Machintosh コンピュータに搭載されているオペレーティング・システム.

スタンス変数に格納されているデータを得るには, アクセスしたいインスタンス変数を内包しているオブジェクトが, データを提供するメソッドを実装していなければならないが, インスタンス変数の有効範囲を指定することで, サブクラスや他のオブジェクトからインスタンス変数に直接アクセスすることも可能である.

- 開発ツールスイート

  Objective-C で開発を行うための開発ツールとして代表的なものに, Xcode が挙げられる.

  Xcode は, “ Apple Inc. が開発, 配布している, Mac/iPhone/iPad/Apple Watch 向けアプリケーション開発用のデベロッパツールセット ”である. ユーザインターフェースのデザイン, コーディング, テスト, デバッグ, Apple Store への提出を行えることによる, スムーズなアプリケーション開発を実現している.

  また, Xcode には, Xcode IDE(統合開発環境:Integrated Development Environment), Swift および Objective-C のプログラミング言語向けコンパイラ, Instruments 分析ツール, iOS Simukator, 最新の iOS/OS X 向け SDK などの機能も用意されている.

- ランタイム環境

  Objective-C では, 可能な限り多くの決定が実行時に行われる. “ オブジェクトの作成やどのメソッドを呼び出すかの決定などの動作は動的に実行される ”ため, Objective-C では, コンパイラだけでなく, コンパイルしたコードを実行するランタイムシステムも必要となる. ランタイムシステムは, Objective-C にとって, 一種のオペレーティングシステムとして動作し, 言語を機能させる.

(B) Objective-C 2.0

Apple Inc. は Mac OS X v10.5 Leopard において, 言語仕様を変更した Objective-C 2.0 を発表した. Objective-C 2.0 では, ガベージコレクションとプロパティの導入, foreach 文の採用, 実装オプションのプロトコル定義の増加, 非公開プライベートメソッドが実装可能, ランタイム構造の変更などが行われた.

2.1.2 OpenCV

OpenCV(Open Source Computer Vision Library)[3] とは, Intel Corporation が開発, 公開しているオープンソースのコンピュータビジョン向けライブラリである. “ 画像処理, 画像解析, 機械学習の機能を持つ C/C++, Java, Python, MATLAB 用ライブラリ ”で, プラットフォームとして Unix/Linux 系 OS, Windows, Android, iOS などをサポートしている. BSD ライセンスで配布されているため, 学術目的のみでなく, 商用目的でも利用することが可能である.

バージョン	リリース年月	公式サポート言語
1.0	2006 年 10 月	C 言語
2.0	2009 年 10 月	C 言語, C++
2.1	2010 年 4 月	C 言語, C++
2.2	2010 年 12 月	C 言語, C++, Python
2.3	2011 年 7 月	C 言語, C++, Python
2.4	2012 年 5 月	C 言語, C++, Python
2.4.1	2012 年 6 月	C 言語, C++, Python
2.4.2	2012 年 7 月	C 言語, C++, Python
2.4.3	2012 年 11 月	C 言語, C++, Python
2.4.4	2013 年 3 月	C 言語, C++, Python, Java
2.4.5	2013 年 4 月	C 言語, C++, Python, Java
2.4.6	2013 年 7 月	C 言語, C++, Python, Java
2.4.7	2013 年 11 月	C 言語, C++, Python, Java
2.4.8	2013 年 12 月	C 言語, C++, Python, Java
2.4.9	2014 年 4 月	C 言語, C++, Python, Java
2.4.10	2014 年 10 月	C 言語, C++, Python, Java
2.4.11	2015 年 2 月	C 言語, C++, Python, Java
3.0	2015 年 6 月	C++, Python, Java(3.0 以降, C 言語はメンテナンス対象外)
3.1	2015 年 12 月	C++, Python, Java

表 2.1: OpenCV のバージョンとリリース年月, 公式サポート言語の一覧

1999 年に開発プロジェクトが開始され, 2000 年のアルファ版公開を皮切りに, 2001 年から 2005 年にかけて, 5 つのベータ版が公開されている. 正式版のリリース後, 2016 年 1 月現在まで開発, 公開されている.(表 2.1 参照.)

最新バージョンである OpenCV 3.x 系列では, C 言語関数形式のインターフェイスがメンテナンス対象外とされ, C++ API を利用することが推奨されている.

(A) OpenCV の機能

OpenCV に実装されている機能 [4] としては, フィルター処理, 変形処理, 構造解析と形状ディスクリプタ, 物体検出, モーション解析と物体追跡, 領域分割, カメラキャリブレーション, 特徴点検出, 機械学習がある.

- フィルター処理

  OpenCV におけるフィルタリングは, 二次元配列である Mat 型で定義された二次元画像に対して処理を実行する. 画像の平滑化や収縮/膨張などの処理を行うことで, 画像のぼかしやノイズキャンセリングを施す.

- 変形処理

  二次元画像の幾何学変換を行い, 変形画像をマッピングする. この処理を施す際は画像の内容そのものは変更されず, ピクセルの座標移動のみを行う..

- 構造解析と形状ディスクリプタ

  二次元画像内に描かれている, あるいは含まれている特定の形状を検出する. 画像内の特定の形状のみを認識したい場合等で利用する.

- 物体検出

  予め設定しているテンプレートと二次元画像とを比較し, テンプレートとマッチングする物体を検出する. 顔認識や指紋認識など, 生体認証等で利用する.

- モーション解析と物体追跡

  動画内で動いている物体の動きを認識する, あるいは動体を追跡する. モーション検知や動体検知等で利用する.

- 領域分割

  二次元画像を形状や色などの特徴を用いて, 複数の領域に分割する. 特定の領域のみで画像処理を施す場合や, 特定の領域のみを抜き出す場合等で利用する.

- カメラキャリブレーション

  カメラ固有の内部パラメータと, ワールド座標系における位置や姿勢を意味する外部パラメータを求める. カメラで撮影した二次元画像の歪みを補正する場合等で利用する.

- 特徴点検出

  二次元画像における特徴点を検出する. 特徴点を検出してパターンマッチングを行う場合等で利用する.

- 機械学習

  OpenCV では, ニューラルネットワークを用いた機械学習を実装することが可能である. 二次元画像を教師データとして入力し, 形状や物体の認識率を向上させる場合等で利用する.

英語	ウクライナ語	トルコ語	タイ語
タガログ語	テルグ語	タミル語	スウェーデン語
スワヒリ語	セルビア語	アルバニア語	スペイン古語
スペイン語	スロベニア語	スロバキア語	ローマ語
ポルトガル語	ポーランド語	ノルウェー語	オランダ語
マレー語	マルタ語	マケドニア語	マラヤーラム語
リトアニア語	ラトビア語	韓国語	カナダ語
イタリア古語	イタリア語	アイスランド語	インドネシア語
チェロキー語	ハンガリー語	クロアチア語	ヒンディー語
ヘブライ語	ガルシア語	中世フランス語	フランス語
フランク語	フィン語	バスク語	エストニア語
エスペラント語	中世英語	グリーク語	ドイツ語
デンマーク語	チェコ語	カタロニア語	ブルガリア語
ブルガリア語	ベンガル語	ベラルーシ語	アゼルバイジャン語
アラビア語	アフリカ語	日本語	中国語 (簡体字)
中国語 (繁体字)	ロシア語	ベトナム語	数学記号

表 2.2: Tesseract-OCR にて認識可能な言語一覧

2.1.3 Tesseract-OCR

Tesseract-OCR[5] とは,“ ヒューレット・パッカード研究所とヒューレット・パッカード社が開発し, 現在は Google が開発, 配布している, 光学文字認識用のオープンソースライブラリ ”である.

ライブラリの大半は C 言語で書かれており, 一部は C++で書かれている. そのため, C 言語コンパイラがインストールされているならば, 様々な環境で動作させる事ができる.

Tesseract-OCR は 2016 年 1 月現在, 64 の言語に対応している.(表 2.2 参照.)

Tesseract-OCR はニューロンネットワークによる機械学習を実装しており, 上記の言語のデータを教師データとして与えることにより, 文字の認識率を向上させることが可能である.

2.2 Web アプリケーション

2.2.1 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,“ 動的なウェブサイト, 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.

アプリケーションフレームワークとは,“ 特定のオペレーティングシステムのためのアプリケーションの標準構造を実装するのに使われるクラスやライブラリの集合 ”である.

つまり, 端的に Web アプリケーションフレームワークを言い換えると, “ Web アプリケーションを開発する際に必要なクラスやライブラリを提供する, Web アプリケーションの土台となるもの ”となる.

HTML や PHP 等の, Web ページ/アプリケーションを開発するための言語を扱うことに長けた専門家が, それらの言語を用いて一から開発したアプリケーションと同等のものが, Web アプリケーションフレームワークを用いることで, 初心者でも開発することが可能となる.

本研究では, Ruby で開発された Web アプリケーションフレームワークで高いシェアを誇る Ruby on Rails を用いて Web アプリケーションを開発する.

2.2.2 Ruby on Rails

Ruby on Rails[6] とは, “ Rails Core Team が開発, 提供しているオープンソースの Web アプリケーションフレームワーク ”である.

オブジェクト指向スクリプト言語である Ruby にて記述されており, オブジェクト指向を強く意識したフレームワークとなっている.

2004 年 10 月に正式版がリリースされてから, 2016 年 1 月現在まで開発, 公開されている.(表 2.3 参照.)

(A) Ruby on Rails の基本理念

Ruby on Rails の特徴は, その開発理念である“ 同じことを繰り返さない (DRY:Don’t Repeat Yourself) ”と’“ 設定より規約 (CoC:Convention over Configuration) ”の順守による, “ 実アプリケーションの開発が他のフレームワークよりも少ないコードで簡単に行える ”点である.

“ 同じことを繰り返さない ”とは, 「定義などの作業は一度のみ行う」, 「重複するコードを記述しない」という意味である.

“ 設定より規約 ”とは, 「慎重に設定された規約に従うことで, 設定を不要にする, あるいは軽減する」という意味である.

これらの基本理念により, Ruby on Rails では, “ コンソール上でコマンドを入力することで, 命名規則に従った複数の関連付けられたファイルが生成されるため, ファイル構成の把握が非常に簡易 ”である.

この Ruby on Rails のコンセプトに影響を受けたフレームワークとして, PHP の CakePHP や Symfony, Perl の Catalyst, Java の Grails といったものが挙げられる.

バージョン	リリース年月
0.8.0	2004 年 10 月
0.9.0	2004 年 12 月
0.10.0	2005 年 2 月
0.11.0	2005 年 3 月
0.12.0	2005 年 4 月
0.13.0	2005 年 7 月
0.14.1	2005 年 10 月
1.0.0	2005 年 12 月
1.1.0	2006 年 3 月
1.2.0	2007 年 1 月
2.0.0	2007 年 12 月
2.1.0	2008 年 5 月
2.2.2	2009 年 11 月
2.3.2	2009 年 5 月
3.0.0	2010 年 8 月
3.1.0	2011 年 8 月
3.2.0	2012 年 1 月
4.0.0	2013 年 6 月
4.1.0	2014 年 4 月
4.2.0	2014 年 12 月
4.2.5	2015 年 11 月

表 2.3: Ruby on Rails のメジャーバージョンとリリース年月の一覧

(B) Ruby on Rails における MVC アーキテクチャ

Ruby on Rails は MVC(Model View Controller) アーキテクチャを採用している.

MVC アーキテクチャとは, アプリケーションの内部データをユーザが直接参照することがないよう, Model, View, Controller の 3 つの要素に分割したものである.

- Model

  アプリケーションが扱う領域のデータと手続きを表現した要素. データの変更をビューに通知するのも Model の役割である.

  Ruby on Rails における Model は, 使用しているデータベースのテーブルごとに Model が用意されている. 利用者からのリクエストで呼びだされたアクションは, Model を介してデータベースとのやり取りを行い, データの取得や新しいデータの格納を行う.

- View

Model のデータを取り出し, ユーザが見るのに適した形で表示する要素. すなわち, データを UI へ出力するのが役割である.

Ruby on Rails における View は, アプリケーション内に複数用意されている. View の 1 つ 1 つが与えられたデータから生成された HTML 文書となっている. アクションに対応する View が 1 つ用意されており, アクションが実行されると, 紐付けされた View が呼び出されて利用者へ返す Web ページを作成する.

- Controller

  ユーザからの入力を Model へのメッセージへと変換して Model へと伝える要素である. すなわち, UI からの入力を担当する.

  Ruby on Rails における Controller は, アプリケーション内に複数用意されている. 1 つ 1 つの Controller は, ユーザの入力をアクションとして紐付けされている Model へと命令する.

2.2.3 PostgreSQL

PostgreSQL[7] とは, “ PostgreSQL Global Development Group がオープンソースで開発, 配布している, オブジェクト関係データベース管理システム (ORDBMS:Objective Relational DataBase Management System) ”である.

1995 年 5 月に正式版がリリースされてから, 2016 年 1 月現在まで開発, 公開されている.(表 2.4 参照.)

PostgreSQL は,“ オープンソースなリレーショナルデータベース管理システム (RDBMS) である Ingres から発展したプロジェクト ”である. PostgreSQL という名は,“ 「Post-Ingres」, つまり Ingres の後継であること ”と, “ SQL データベース操作構文を用いてデータベース操作が可能であること ”が, その名の由来となっている. Ingres の課題であった, 「ユーザが新たな定義域を既存の単純な定義域から定義できない」という実装の限界に対処することを目的として開発された.

データベースの操作には SQL データベース操作構文を利用しているが, PL/pgSQL, PL/PSM, スクリプト言語 (PL/Perl, PL/php, PL/Python, PL/Ruby, PL/Tcl, PL/Lua), コンパイラ言語 (C 言語, PL/Java), 統計処理言語 (PL/R) の関数を実行することも可能である.

(A) リレーショナルデータベース (Relational DataBase)

リレーショナルデータベース [8] とは,“ 一見のデータを複数の属性 (カラム) の値 (フィールド) の組 (レコード) として表現し, 組を列挙することでデータを格納していく方式のデータベース ”のことである. 属性を列, 組を業とする表 (テーブル) の形で表されることが多い. 複数のテーブルに含まれる同じカラムを関連付けることができ, 複雑なデータや大規模なデータを柔軟に取り扱うことができる.

データベースの構造では最も普及しており, 単にデータベースといった場合はリレーショナルデータベースであることが多い.

バージョン	リリース年月
0.01	1995 年 5 月
1.0	1995 年 9 月
6.0	1997 年 1 月
6.1	1997 年 6 月
6.2	1997 年 10 月
6.3	1998 年 3 月
6.4	1998 年 10 月
6.5	1999 年 6 月
7.0	2000 年 5 月
7.1	2001 年 4 月
7.2	2002 年 2 月
7.2	2002 年 11 月
7.4	2003 年 11 月
8.0	2005 年 1 月
8.1	2005 年 11 月
8.2	2006 年 12 月
8.3	2008 年 2 月
8.4	2009 年 7 月
9.0	2010 年 9 月
9.1	2011 年 9 月
9.2	2012 年 9 月
9.3	2013 年 9 月
9.4	2014 年 12 月
9.5	2016 年 1 月

表 2.4: PostgreSQL のメジャーバージョンとリリース年月の一覧

(B) Relational DataBase Management System

Relational DataBase Management System(RDBMS)[9] とは, “ リレーショナルデータベースを管理するための専用ソフトウェア ”である.

ストレージ内に専用の管理領域を確保し, テーブルの作成や消去, 構造の修正, データの追加, 検索, 抽出, 修正, 削除等を行う. データベース管理者や利用者が直接操作を行う他に, 外部のソフトウェアから接続を受け付け, ソフトウェアのプログラムから操作を行うことも可能である. RDBMS への照会や操作の指示には, SQL と呼ばれるデータベース操作言語が標準として広く利用されている.

RDBMS は, “ 不正なデータの記録を拒否するなどしてデータベースの整合性を保つ ”, “ 権限のない利用者による不正な読み出しや改ざん等からデータを保護する ”等の仕組みを持つ. また, 関連する複数の処理を一体化して矛盾なく実行するトランザクション処理

を行ったり, 障害に備えてデータベースのバックアップを行い, 破損時には過去のある時点の状態へと復旧したりといった機能を備えたものもある.

2.2.4 Heroku

Heroku[10] とは,“ 2007 年に創業された同名の企業が開発と運営を行っている, PaaS(Platform as a Service) ”である.

今回は, Ruby on Rails で作成した Web アプリケーションを Heroku サーバにて動作させ, インターネットに公開するために利用する.

(A) PaaS

PaaS[11] とは, “ アプリケーションを実行するためのプラットフォーム ”である.

“ アプリケーションを実行するためのプラットフォーム ”は, 「ハードウェア」, 「ネットワーク」, 「仮想化環境」, 「オペレーティングシステム」, 「データベース」, 「アプリケーションフレームワーク」で構成されている.

Web アプリケーションを自らの手で一から全て開発する場合,

1. サーバ PC やルータ等のハードウェアを購入
2. それらをインターネットに接続し, かつプライベート空間を分離するためのファイアウォールを含むネットワークを構築
3. サーバの仮想化環境を整備
4. Linux や Windows サーバ等のオペレーティングシステムをインストール
5. Oracle や MySQL, PostgreSQL 等のデータベースをセットアップ
6. Java や Ruby, PHP 等のアプリケーション実行環境をセットアップ

という手順を踏んで, アプリケーションを公開するための環境を整えなければならない.

PaaS は, これら“ プラットフォームの構築にかかる様々な作業を代行してくれるサービス ”である. PaaS を利用することで, 最初に行わなければならない環境構築への労力を軽減し, また, プラットフォームの保守運用にかかる費用を削減することが可能となる.

Heroku は自社の目的として, “ 技術者をビジネスの本質的な価値提供 (アプリケーション開発) に集中させること ”, “ 開発者の生産性を最大化すること ”を掲げており, PaaS の提供だけでなく, 高頻度の機能追加やアドオンの提供を行っている.

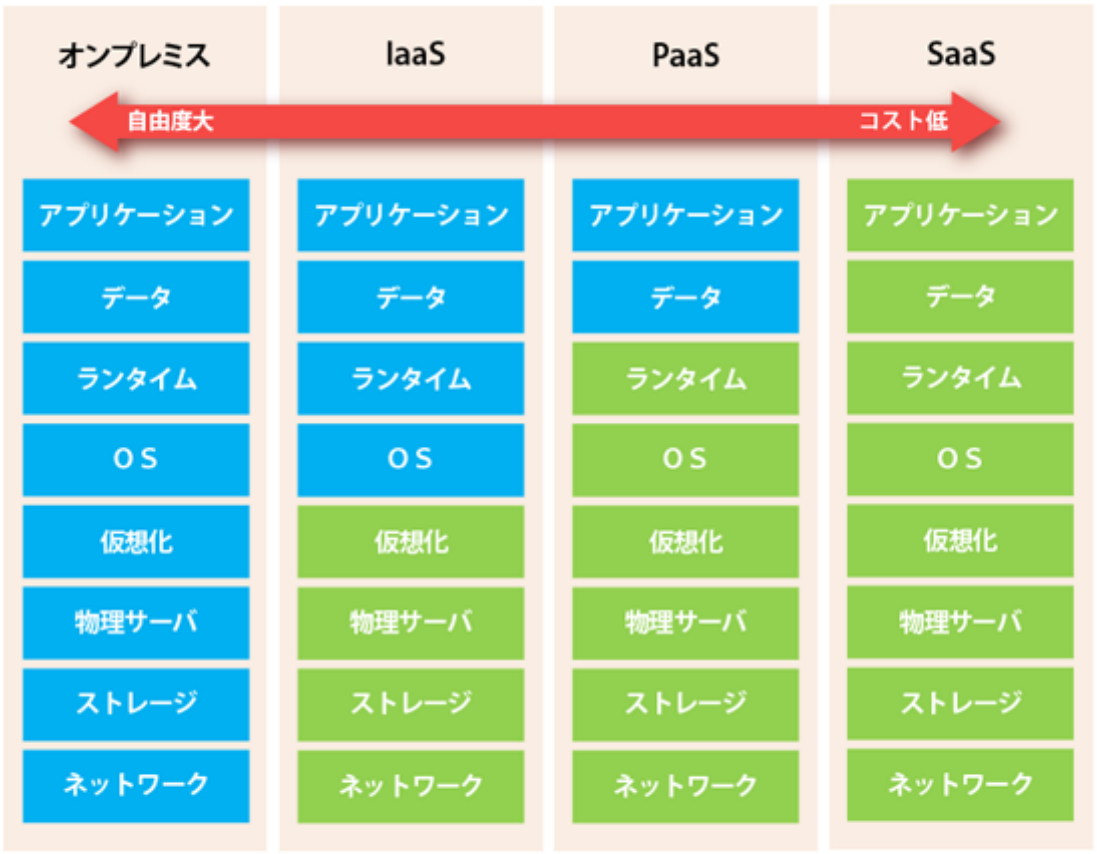

図 2.1: オンプレミス, IaaS, PaaS, SaaS の比較 (出典：CodeZine『PaaS の基礎知識と Heroku で開発を始める準備』)

(B) オンプレミス, IaaS, SaaS との違い

Web アプリケーション開発の補助を行うサービスとして, 「IaaS(Infrastructure as a Service)」, 「SaaS(Software as a Service)」があり, そして全てを自ら管理する「オンプレミス」がある.(図 2.1 参照.)

- オンプレミス

  プラットフォームの全てを自身で管理する.

  「ハードウェア」, 「ネットワーク」, 「仮想化環境」, 「オペレーティングシステム」, 「データベース」, 「アプリケーションフレームワーク」を自身で選択できるため, 開発の自由度が高い.

  しかし, 環境構築からアプリケーション公開までの労力は必然的に重たくなる.

- IaaS

  プラットフォームのうち, 「ハードウェア」, 「ネットワーク」, 「仮想化環境」までをサービス事業者が管理する.

  「オペレーションシステム」, 「データベース」, 「アプリケーションフレームワーク」は自ら選択できるため, サービス利用者の慣れ親しんだ環境でのアプリケーション開発を行うことができる.

- PaaS

  プラットフォームの「ハードウェア」, 「ネットワーク」, 「仮想化環境」, 「オペレーティングシステム」, 「データベース」までをサービス事業者が管理する.

  「アプリケーションフレームワーク」にて開発したアプリケーションをサービス会社のサーバにアップロードすることで, サービス事業者がランタイム (アプリケーションの実行) まで行う.

  サービス利用者はアプリケーションの開発, 運用のみに注力すればよいが, 開発環境の自由度は IaaS よりも低い.

- SaaS

  プラットフォームの全てをサービス事業者が管理する.

  管理コストは最も低いが, 開発環境の自由度も低い.

  サービス事業者が提供するソフトウェアをインストールして利用する形態を採用しているため, サービス事業者によって開発可能なサービスが異なる可能性がある.

2.3 VPS サーバ

本研究では, 開発した Web アプリケーションを公開するにあたって, 琉球大学工学部情報工学科が管理している VPS サーバを利用した.

2.3.1 CentOS

CentOS とは,“ The CentOS Project がオープンソースライセンスで開発, 配布している Linix ディストリビューション ”である. CentOS という名称は「コミュニティベースで開発された, 有償版に匹敵するオペレーティングシステム (Community ENTerprise Operating System)」が由来となっている.

2004 年 5 月のリリースから, RHEL の最新版が公開される度に, その後を追うようにしてメジャーアップデートを重ね, 2016 年 1 月現在まで開発, 配布されている.

“ Red Hat Enterprise Linux(RHEL)[3] の完全互換を目指して開発 ”され, Red Hat 社がオープンソースにて公開している RHEL のソースコードを基に,“ Red Hat 社の商標や商用パッケージを除去し, リビルドしたもの ”が CentOS である. このことから, CentOS は一般に“ RHEL クローン ”[4] と呼称されることもある.

Linux をベースとしたオペレーティングシステムであること, GNU ライセンスによる配布による低コストでの導入の容易さから, 主に中小規模のプロジェクトにて採用されることが多い.

[3]Red Hat 社が開発, 販売している業務向け Linux ディストリビューション.

[4]RHEL を基とした Linux ディストリビューション全般を含んだ呼称であるため, White Box Enterprise Linux や Scientific Linux も RHEL クローンに含まれる.

第3章　実装

3.1　システムの流れ

本研究にて構築するシステムは, 主に「iOS アプリケーション」, 「Web アプリケーション」, 「データベース」の三要素にて成り立つ.(図 3.1 参照.)

3.2　iOS アプリケーション

iOS アプリケーションを作成するにあたり, 以下の言語とライブラリ, 統合開発環境を利用した.(表 3.1 参照.)

3.2.1　要件定義

iOS アプリケーションには, 以下の機能を実装する.

- iOS デバイスのカメラにて撮影する
- 撮影画像から目的の文字列を取得する
- 取得した文字列を Web アプリケーションのデータベースに POST する.

これらの要件を, UML を用いて分析する.

3.2.2　ユースケース

ユーザが iOS デバイスのカメラを用いて撮影した画像から, 文字列を検出する. その文字列を Web アプリケーションのデータベースに送信し, 集積する.

Xcode	
Objective-C 2.0	
Tesseract-OCR	
OpenCV	

表 3.1: iOS アプリケーション作成時に利用した環境のバージョン一覧

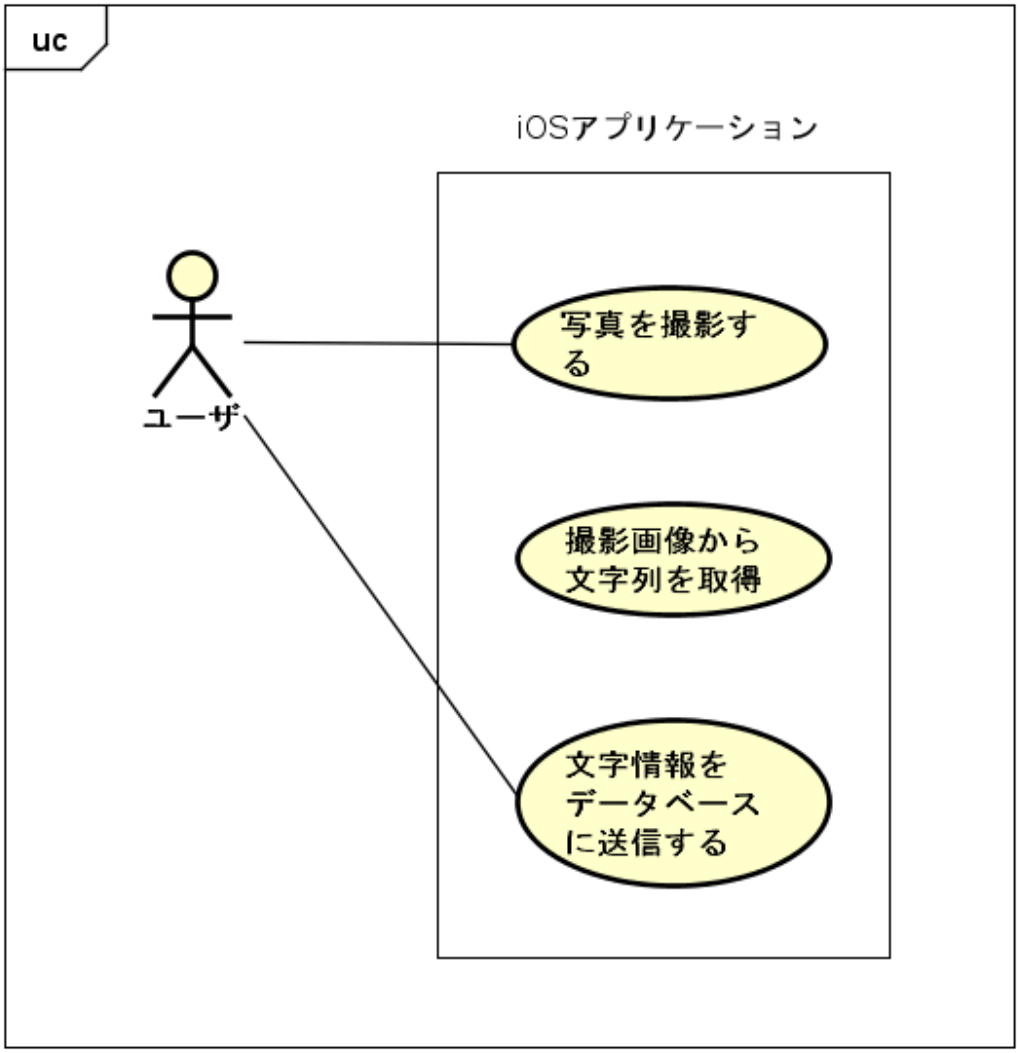

図 3.1: iOS アプリケーションのユースケース図

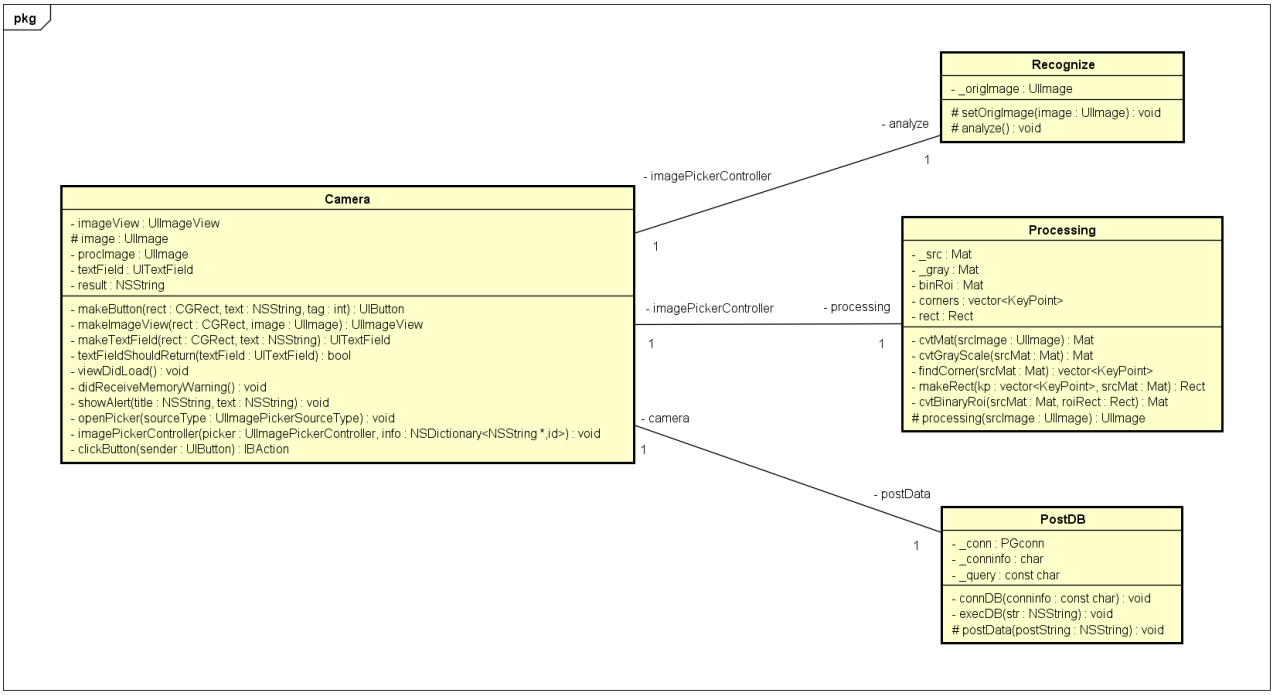

図 3.2: iOS アプリケーションのクラス図

3.2.3 クラス

(A) Camera クラス

Camera クラスでは,「iOS アプリケーションの MainView にボタン等を配置する」,「iOS デバイスのカメラを起動し, 撮影した画像を保存する」,「他クラスのメソッドを呼び出す」ことが可能である.

- 属性
  - imageView：UIImageView
    撮影した画像を出力するためのローカル変数. 出力画像のサイズ等を決定する.
  - image：UIImage
    撮影した画像を格納するための変数.
  - procImage：UIImage
    Processing クラスにて画像処理された画像を格納するための変数. procImage に格納されている画像を用いて, 光学的文字認識を行う.
  - textField：UITextField
    Recognize にて取得した文字列を表示するテキストボックス.

- result：NSString
  Recognize にて取得してきた文字列を格納するためのローカル変数.

- 操作

  - makeButton：UIButton
    iOS アプリケーションの MainView にボタンを作成するメソッド.
    ボタンを配置する座標, サイズ, ボタンに書かれる文字列, ボタンに付加するタグを指定して呼び出すことで, iOS アプリケーションの MainView にボタンを作成し配置する.
    * rect：CGRect
      ボタンの座標, サイズを決定する変数.
    * text：NSString
      ボタンに書かれる文字列を決定する変数.
    * tag：int
      ボタンに整数型のタグを付加するための変数.
  - makeImageView：UIImageView
    iOS アプリケーションの MainView に画像を表示するための SubView を作成するメソッド.
    画像を配置する座標, サイズ, 出力する画像を指定して呼び出すことで, iOS アプリケーションの MainView に画像を出力する.
    * rect：CGRect
      画像を出力する SubView の座標, サイズを指定する変数.
    * image：UIImage
      SubView に出力する画像を指定する変数.
  - makeTextField：UITextField
    iOS アプリケーションの MainView にテキストフィールドを作成するメソッド.
    テキストフィールドを配置する座標, サイズ, 入力されるテキストを指定して呼び出すことで, iOS アプリケーションの MainView にテキストフィールドを作成し配置する.
    * rect：CGRect
      テキストフィールドを配置する座標, サイズを指定する変数.
    * text：NSString
      テキストフィールドに入力する文字列を指定する変数.

- textFieldShouldReturn：BOOL

  テキストフィールドにて, キーボードでリターンキーを押した際の挙動を制御するメソッド.

  文字列を編集後にリターンキーを押した際に呼び出され, SubView として呼びだされているキーボードを引っ込める.

  * textField：UITextField
    編集したい文字列が入力されているテキストフィールド.

- viewDidLoad：void

  アプリケーションが起動した際に一度だけ読み込まれるメソッド.

  変数の初期化や MainView にボタン等を配置する際に利用する.

- didReceiveMemoryWarning：void

  iOS アプリケーション使用時に, メモリが不足する際に呼び出されるメソッド.

  すべての View に対して参照を開放する際に利用する.

- showAlert：void

  アラートビューを表示するメソッド.

  アラートビューのタイトルと表示するアラートの内容を指定して呼び出すことで, iOS アプリケーションの MainView 上に SubView としてアラートが表示される.

  * title：NSString
    アラートビューのタイトルを指定する変数.
  * text：NSString
    アラートビューに表示するテキストを指定する変数.

- openPicker：void

  iOS アプリケーションにて, 画像を取得するためのメソッド.

  カメラ, フォトアルバム, フォトライブラリから画像を取得する.

  このメソッドでは, iOS のカメラ機能にて撮影した画像を取得する際に呼び出す.

  * sourceType：UIImagePickerSourceType
    イメージピッカーの属性を決定する変数. カメラ, フォトアルバム, フォトライブラリのいずれかから選択する.

- imag：PickerController:void

  画像取得後に呼び出されるメソッド.

  画像を取得した後の処理を行う他クラスのメソッドを呼び出す.

  * picker：UIImagePickerController
    画像取得元のイメージピッカーを指定する変数.

    * info：NSDictionary<NSString *, id>
      取得した画像の情報を格納する変数.

  – clickButton：IBAction
    iOS アプリケーション上の MainView に配置されているボタンを押下した際に呼び出されるメソッド.

    * sender：UIButton
      どのボタンが押下されたのかを決定する変数.

(B) Processing クラス

Processing クラスでは,「撮影画像を OpenCV にて加工可能となるように処理を施す」,「撮影画像中の文字列が含まれている箇所に注目する」,「注目している箇所のみで光学的文字認識が施されるように, 撮影画像を加工する」ことが可能である.

- 属性

  – _src：Mat
    入力画像を指定する変数.

  – _gray：Mat
    グレースケール変換された画像を格納する変数.

  – binRoi：Mat
    バイナリスケール変換された画像を格納する変数.

  – corners：vector<KeyPoint>
    検出した特徴点を格納する変数.

  – rect：Rect
    特徴点を内包した矩形の座標, サイズ決定する変数.

- 操作

  – cvtMat：Mat
    UIImage 型の画像を Mat 型に変換するメソッド.
    UIImage 型の入力画像を指定して呼び出すことで, OpenCV で処理可能な Mat 型に変換する.

    * srcImage：UIImage
      UIImage 型の画像を格納する変数.

- cvtGrayScale：Mat

  Mat 型画像をグレースケール化するメソッド.

  cvtMat にて Mat 型に変換した入力画像にグレースケール化処理を施す.

  * srcMat：Mat
    3 チャネルの入力カラー画像を格納する変数.

- findCorner：vector<KeyPoint>

  グレースケール画像から特徴点を検出するメソッド.

  cvtGrayScale にてグレースケール処理を施した画像を指定することで, その画像中から特徴点を検出する.

  * srcMat：Mat
    グレースケール画像を格納する変数.

- makeRect：Rect

  特徴点を内包する矩形の座標, サイズを決定するメソッド.

  特徴点の集合を指定することで, 特徴点の多い部分に注目するための矩形を設定する.

  * kp：vector<KeyPoint>
    特徴点の集合を格納する変数.
  * srcMat：Mat
    グレースケール画像を格納するための変数.

- cvtBinaryRoi：Mat

  Mat 型カラー画像をバイナリ画像へ変換するメソッド.

  makeRect にて設定した矩形を指定することで, 注目している範囲をバイナリ画像化する.

  * srcMat：Mat
    3 チャネルの入力カラー画像を格納する変数.
  * roiRect：Rect
    注目する矩形範囲を決定する変数.

- processing

  Processing クラスのメソッドをコントロールするメソッド.

  * srcImage：UIImage
    Camera クラスの openPickerController メソッドにて取得した撮影画像を格納する変数.

(C) Recognize クラス

Recognize クラスでは,「光学的文字認識を実行する」,「取得した文字列を Camera クラスに渡す」ことが可能である.

- 属性
  - _origImage：UIImage
    光学的文字認識を実行する画像を格納するための変数.
- 操作
  - setOrigImage：void
    光学的文字認識を実行する画像を取得するためのメソッド.
    Processing クラスの processing メソッドにて処理の施された画像を受け取り,_origImage に格納する.
    - image：UIImage
      光学的文字認識を施す画像を指定する変数.
  - analyze：void
    Tesseract-OCR による光学的文字認識を実行するメソッド.

(D) PostDB クラス

- 属性
  - _conn：PGconn
    PostgreSQL データベースとの接続状態を格納する構造体.
  - _conninfo：const char
    PostgreSQL データベースの接続する際に必要となる情報を格納する変数.
  - _query：char
    接続した PostgreSQL にて実行するクエリを決定する変数.
- 操作
  - connDB：void
    Web アプリケーションの PostgreSQL に接続するメソッド.
    - conninfo：const char
      PostgreSQL への接続状態を保存する変数.

rbenv	
Ruby	
RubyGems	
Ruby on Rails	
PostgreSQL	
Heroku	

表 3.2: Web アプリケーション作成時に利用した環境のバージョン一覧

- execDB：void
  Web アプリケーションの PostgreSQL データベースへクエリを発行し, データベースへデータを追加する.
  - str：NSString
    PostgreSQL データベースへ送信する文字列を決定する変数.
- postData：void
  PostDB クラスのメソッドをコントロールするメソッド.
  - postString：NSString
    Web アプリケーションの PostgreSQL データベースへ送信する文字列を決定する変数.

3.3 Web アプリケーション

Web アプリケーションを作成するにあたり, 以下の言語とライブラリ, サービスを利用した.(表 3.2 参照.)

3.3.1 要件定義

Web アプリケーションには, 以下の機能を実装する.

- データベースにて, iOS アプリケーションから送信された文字列を送信された日時/時刻とともに管理する.
- 集積したデータを文字列, 日付にて検索する.

これらの要件を, UML を用いて分析する.

3.3.2 ユースケース

ユーザは iOS アプリケーションから送信された全データを閲覧することができ, 文字列, 送信した日付で検索することができる.

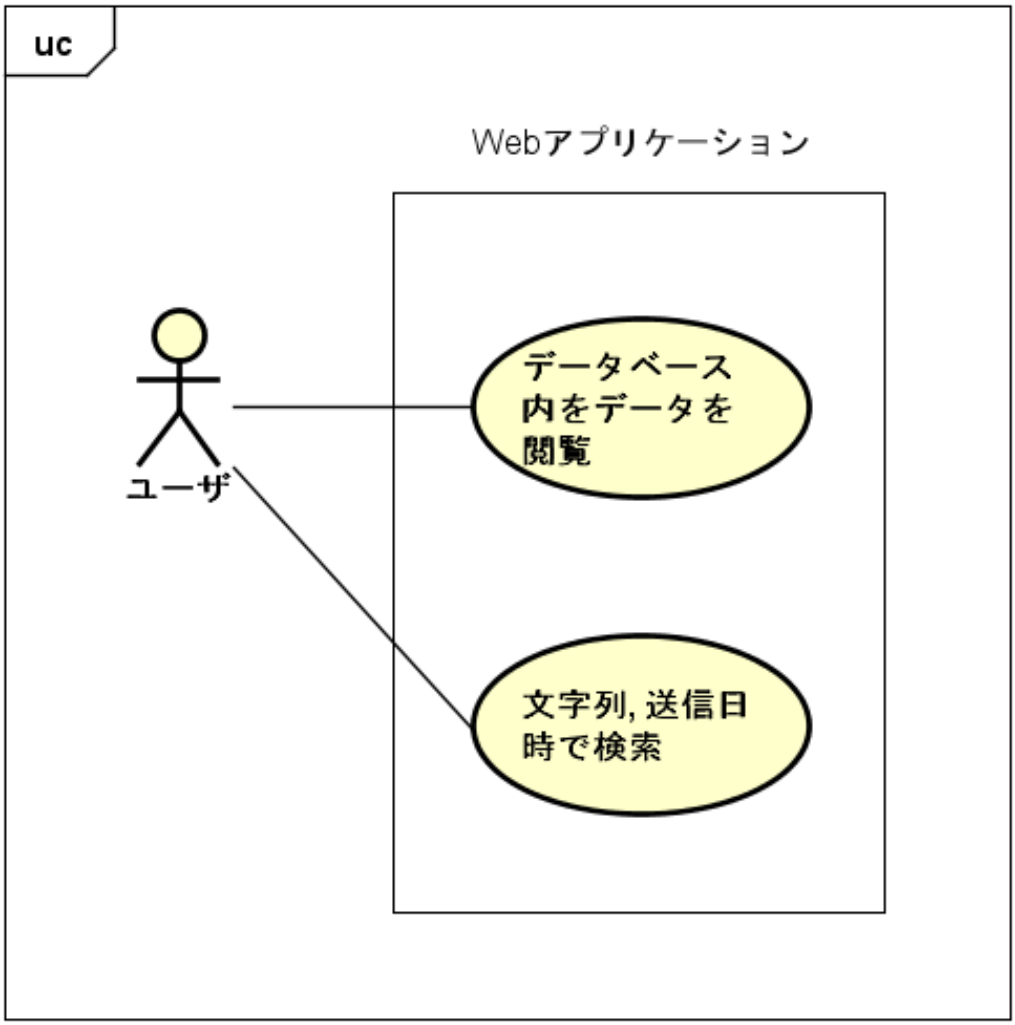

図 3.3: Web アプリケーションのユースケース図

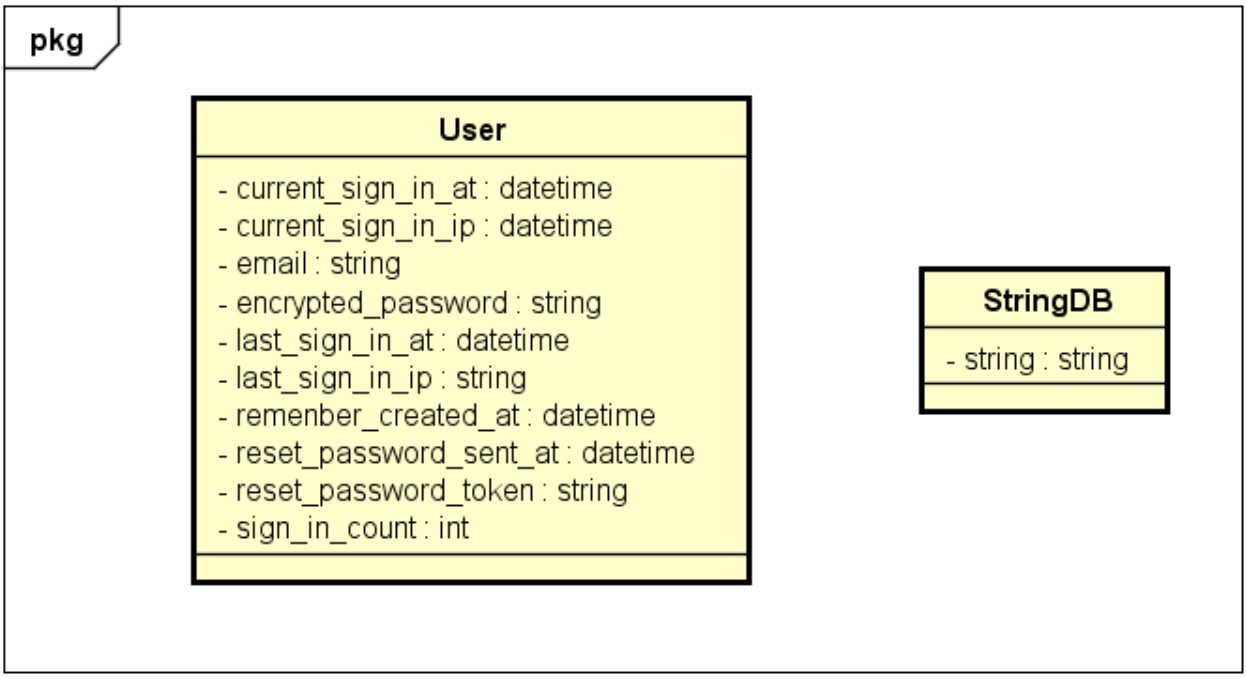

図 3.4: Web アプリケーションのクラス図

3.3.3 クラス

(A) User クラス

- 属性
  - current_sign_in_at：datetime
    Web アプリケーションにユーザ登録したユーザがサインインした時刻を記録するカラム.
  - current_sign_in_ip：datetime
    ユーザがサインインした際のリモート IP を記録するカラム.
  - email:string
    ユーザのサインイン時に利用する E メールアドレスを記録するカラム.
  - last_sign_in_at：datetime
    ユーザが最終サインインした時刻を記録するカラム.
  - last_sign_in_ip：datetime
    ユーザが最終サインインした際のリモート IP を記録するカラム.
  - remenber_created_at：datetime
    ユーザ登録した時刻を記録するカラム.

- reset_password_sent_at：datetime
  パスワードをリセットする操作を行った際の時刻を記録するカラム.
- reset_password_token：string
  パスワードをリセットする操作を行った際に設定した新しいパスワードを記録するカラム.
- sign_in_count：int
  ユーザがサインインした回数を記録するカラム.

(B) StringDB クラス

- 属性
  - string：string
    光学的文字認識にて取得した文字列を記録するカラム.

第4章　本研究の利用例, 利用アイデア

本研究では, 「iOS のポータビリティの高さ」と「沖縄県の観光産業」に注目し, 観光地の文字情報をデータベースに集積することを行った.

しかし, 本研究は上記の目的のみにとどまらず, iOS デバイスのポータビリティを活かし, 様々な分野に派生することが可能であると考える.

4.1　利用アイデア１：緊急車両や警察車両への住所通達システム

4.1.1　背景と目的

本研究にて構築したシステムは, スマートデバイスで撮影した画像から文字列を取得し, Web アプリケーションのデータベースに送信する, というものである.

ここでは, 画像に“ GPS による位置情報 ”を付与する.

SNS が急速に普及して以降, 事件や事故が起きた際に, その現場周辺にいる人々がその様子を撮影し, SNS にその時の状況をアップロードする, ということが行われるようになっている. ここでは, その“ 画像による状況把握の容易さ ”と, “ スマートデバイスにて GPS の位置情報が利用可能である ”ことに注目する.

4.1.2　システムの要件定義

ユーザが現場周辺でスマートデバイスを用いて画像を撮影する. その画像に GPS の位置情報とどの緊急車両 (救急, 消防, 警察) が必要であるかの情報を付与し, Web アプリケーションに送信する.

受信した Web アプリケーションは, それらの情報を受け取ると各緊急機関に現場画像と位置情報, どの緊急車両を向かわせるか要請する.

そして, 緊急車両は指定された車両で GPS による現場の位置情報を元に, 現場に向かう.

4.1.3　このシステムを利用することで解決する問題

このシステムを利用することで解決する問題は,

- 緊急車両を呼ぶ際に, 現場の住所がすぐに分からない

  現場の住所を電話口で職員に伝える際, 通報者の近くに現住所を示すものがない場合, 正確な住所を伝えることが困難となることも考えられる.

  このシステムでは, スマートデバイスから取得した GPS の位置情報を Web アプリケーションに送信するため, 住所が不明な場所においても, 正確な現場の位置を伝えることが可能となる.

- 偽情報による緊急車両の出動

  通報による緊急車両の出動は, 故意な偽情報によって行われることがある. それらの悪意ある行動による緊急車両の不必要な出動は, 真に必要な現場への出動が遅延する可能性を生む.

  このシステムでは, スマートデバイスのポータビリティと情報発信力を活用しており, 大規模な事件, 事故の場合は複数人が通報システムを利用することが考えられるため, 偽情報による緊急車両の出動を抑制することが可能だと推測できる.

4.2

4.2.1

参考文献

[1] http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h25/html/nc243110.html

[2] https://ja.wikipedia.org/wiki/Objective-C

[3] https://developer.apple.com/jp/documentation/ObjC.pdf

[4] https://developer.apple.com/support/xcode/jp/

[5] http://www.buildinsider.net/small/opencv/001

[6] http://www.apple.com/jp/osx/continuity/

[7] http://allabout.co.jp/gm/gc/3588/

[8] http://news.mynavi.jp/news/2015/08/06/144/

[9] http://windows.microsoft.com/ja-jp/windows/history#T1=era0

[10] https://en.wikipedia.org/wiki/List_of_Microsoft_Windows_versions

[11] http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1

[12] https://github.com/tesseract-ocr/tesseract/blob/master/README.md

[13] https://ja.wikipedia.org/wiki/Model_View_Controller

[14] http://www.rubylife.jp/rails/ini/index7.html

[15] http://codezine.jp/article/detail/8051

[16] http://e-words.jp/w/RDBMS.html

[17] https://en.wikipedia.org/wiki/CentOS

[18] http://japan.zdnet.com/article/35067916/

謝辞

本研究の遂行、また本論文作成にあたり、御多忙にもかかわらず終始懇切なる御指導と御教授を賜りました谷口祐治准教授に深く感謝致します。

また、一年間共に研究を行い、温かな気遣いと励ましをもって支えてくれた学習環境システム研究室の山本耀悟君、大濱ゆめさん、ならびに琉球大学工学部情報工学科インターネットシステム研究室の皆様に感謝致します。

最後に、有意義な時間を共に過ごした情報工学科の学友、ならびに物心両面で支えてくれた両親に深く感謝致します。

平成 28 年 2 月 6 日

長倉貴洋