CO-006

複数種の画像認識エンジンを統合するフレームワークの提案とスマートフォン向け案内サービスへの適用

Proposal of a framework integrating multiple types of image recognition engines and its application to information service for smartphones

渋沢 潮‡ 鈴木 督史‡ 中村 幸博‡ 山下 遼‡ 茂木 学‡ 手塚 博久‡ 木下 真吾‡
Ushio Shibusawa Masafumi Suzuki Yukihiro Nakamura Ryo Yamashita Manabu Motegi Hirohisa Teduka Shingo Kinoshita

1. はじめに

スマートフォンの高性能化と無線ネットワークの高速化を背景に，簡易な情報取得手段として画像や動画の活用が注目されている．現在までに，物体認識技術，電子透かし技術，文字認識技術，バーコード認識等，カメラで取得した画像や動画から対象物の ID や埋め込まれた ID を認識する技術（以降，認識技術と呼ぶ）が実用段階に入りつつある．そして，観光，交通，生活支援をはじめとする様々な分野で，それら認識技術を活用したスマートフォンやタブレット等のカメラ付き携帯端末向けサービスが実現されている．

ここでの次の課題は，これら認識技術の更なる高度化に加え，それら異なる認識技術を統合する技術の確立が重要と考える．すなわち，それら認識技術の統合により相互補完が図れる．これにより，認識性能の向上や認識対象のアイテムの拡大が見込め，より実用性の高いサービスの創出が期待できる． 観光地，ショッピングモール，駅，空港，展示会等の我々の生活環境には，物体認識技術が得意とする商品や看板，オブジェ，2 次元バーコードや静止画透かしを埋め込み可能なポスターやパンフレット，動画透かしが埋め込み可能なデジタルサイネージ等，様々なアイテムがある．統合技術の実現により，これら場所における様々なシーンで，多様なアイテムを対象とした情報案内サービスを利用できるようになり，ユーザの利便性を大きく向上できる．以降，異なる認識技術を統合する技術をユニバーサルオブジェクト認識（Universal Object Recognition，以下 UOR）と呼ぶ．

本論文では，個々の認識技術を実装したソフトウェア（以降，認識エンジンと呼ぶ）を統合し，様々なアイテムを対象とした携帯端末向け情報案内サービスの実現を狙い，(1)認識エンジンを統合したアプリケーション開発を効率的に行える階層構造，(2)インタフェースや処理性能等が異なる認識エンジンの仮想化，(3)サーバと携帯端末の認識エンジンを統合する統合方式を特徴とした UOR フレームワークを提案する．また，展示パネル，展示オブジェ，デジタルサイネージ等の様々なアイテムが用いられる展示会を対象に，UOR フレームワークに基づいた展示案内サービスを具現化し，展示案内サービスとフレームワークの有効性を検証する．

2. 関連研究

これまで，Mobile Visual Search の分野などで，携帯端末に内蔵されたカメラで撮影したアイテムを識別し，関連する情報を提示する様々なシステムが提案されている[1-4]．また，本論文で扱う異なる画像認識技術を組み合わせたシステムも多く提案されている．たとえば，物体認識技術と文字認識技術を統合して文字認識や物体認識の精度を向上させる試み[5-9]や，異なる画像認識技術で対象を識別した結果を統合し，対象の認識精度を向上させるシステム[10]などがある．また，物体認識機能と移動体のトラッキング機能をサーバとクライアントに配置したハイブリッドシステムも提案されている[11]．これらは異なる認識技術の統合により相互補完を図り，認識性能の向上や新たな認識機能の実現を目指したものと言える．しかしながら，これらは本論文で提案する複数の認識エンジンを統合するフレームワークの検討までは行っていない．

商用サービスとして画像認識 API がクラウドサービスとして提供されつつある[12-14]．これらクラウドサービスは，一般物体認識や顔検出等様々な認識エンジンを Web API で利用可能である．特定のクラウドサービスを用いてアプリケーションを開発するのであれば，非常に効率的な開発が望める．しかしながら，他の認識エンジンも活用してアプリケーション開発をする場合には，それらとの統合の仕組みが求められる．

3. 異なる認識エンジン統合の課題

本章では，市中で利用可能な様々な認識エンジンを統合するための課題を，認識エンジンのインタフェース，配置，スケーラビリティの観点から整理する．

(1) 認識エンジン毎に異なるインタフェース

個々の認識エンジンでは，コマンド名，パラメータの内容，入出力データのファイル形式やフォーマット等が API として規定されているものの，これらは認識エンジン毎に異なる．さらに，一部のパラメータの調整は認識技術に関する専門知識を必要とする．また，個々の認識エンジンでは，識別するアイテムの ID がエンジン固有の ID 体系で管理されているケースが多い．例えば，商品には JAN コード等の規格化されたバーコードがパッケージに印刷されている．これをバーコード認識エンジンと物体認識エンジンで認識した場合，同一アイテムに対してバーコード ID と物体認識エンジン固有の ID の 2 種類が出力される．それゆえ，異なる認識エンジンを統合するためには，認識技術の専門知識を駆使しながら，これら API とアイテムの ID 体系の違いを考慮して実装をする必要があり，多大な手間を要する．

(2) 認識エンジンの配置

我々が利用できる認識エンジンには Google Cloud Vision API[12]等のようにクラウド上やサーバ上で動作させることを想定した認識エンジンだけでなく，携帯端末に組み込む

ことを想定した認識エンジンもある．それゆえ，異なる認識エンジンを統合するためには，それらの違いを考慮した実装が必要となる．

(3) 認識エンジン毎に異なるスケーラビリティ

認識エンジンには，クラウドサービス[12-14]で提供されているようなスケーラビリティを考慮した認識エンジンだけなく，ライブラリ等の認識モジュールのみが提供され，スケーラビリティまでは考慮していないケースもある．それゆえ，異なる認識エンジンを統合するためには，個々の性能に応じて多重化を図り，スケーラビリティを確保することが重要となる．

以上のように，異なる認識エンジンを統合したアプリケーションを実現するためには，開発者は，個々の認識技術の特徴を理解するだけでなく，個々の認識エンジンの API や ID 体系，配置，処理性能の違いを考慮してアプリケーションを開発する必要がある．それゆえ，単一の認識エンジンを対象としたアプリケーション開発に比べて，多大な開発コストを要する．

4. ユニバーサルオブジェクト認識フレームワークの提案

4.1 基本構造

図 1 に異なる認識エンジンを統合するための UOR フレームワークの基本構造を示す．

このフレームワークでは，システム全体を，(1)エンジン実装レイヤ，(2)シナリオ実装レイヤ，(3) UI 実装レイヤで構成する．エンジン実装レイヤでは，認識エンジンを仮想化し（以降，仮想認識エンジンと呼ぶ），認識エンジンの API と ID 体系の抽象化，処理性能の違いに伴う多重化，種類の異なる認識エンジンの簡易な追加を実現する．シナリオ実装レイヤでは，仮想認識エンジンの API を用いて異なる認識エンジン統合のロジック（シナリオ）を実装する．UI 実装レイヤではユーザインタフェースから実行するシナリオを呼び出し，認識結果に基づいて，アイテムに関する情報を提示する．

このような分離のねらいは，認識技術の専門知識を要する実装範囲を限定し，認識エンジンを統合したアプリケーション開発を見通し良く行うとともに，認識エンジンの多重化や種類の追加，統合ロジックの追加変更に対して，システム全体の修正を最小限に抑えて実現することにある．

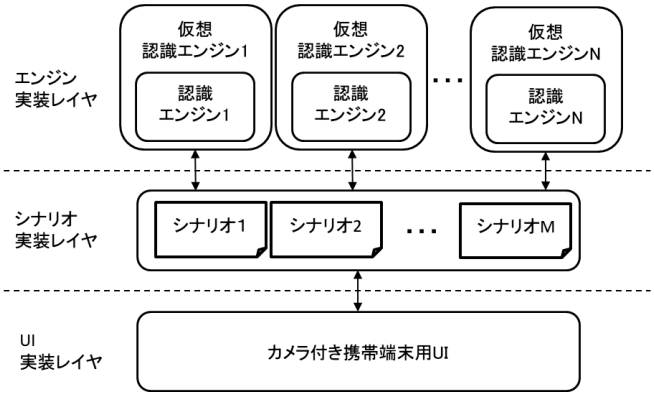

図 1　UOR フレームワークの基本構造

4.2 エンジン実装レイヤ

エンジン実装レイヤの構造を図 2 に示す．このレイヤは，アダプタとオブジェクト ID マネージャによる認識エンジンの API と ID 体系の抽象化を実現する．また，ジョブキューによる処理性能の違いに起因する多重化や種類の異なる認識エンジンの簡易な追加を実現する． 以降ではそれぞれについて詳細を述べる．

4.2.1 アダプタとオブジェクト ID マネージャによる認識エンジン API の抽象化と ID の共通化

各アダプタは認識エンジン毎の詳細な API を簡易な形式に抽象化する．具体的には，認識エンジン毎の独自 API を用いて，認識処理を行って認識結果を得るとともに，抽象化 API を用いて，シナリオ実装レイヤから認識実行要求を取得し認識結果を返却する．

抽象化 API の認識実行要求のパラメータは，処理対象となる画像ファイルの ID 等で構成する．また，認識結果のパラメータは，認識したアイテムの ID，画像内のアイテムの座標値等の共通的な要素で構成し，認識エンジン固有のパラメータ等は扱わない．

オブジェクト ID マネージャは，認識エンジン毎のアイテムの ID 体系を共通化する．具体的には，認識エンジン毎に異なるアイテムの ID 体系を共通の ID に変換する対応表を持ち，認識したアイテムの ID を共通の ID（オブジェクト ID）に変換する．

4.2.2 ジョブキューによる認識エンジンの追加と多重化

認識エンジンには，サーバ側で動作する認識エンジンと携帯端末上（以降，クライアントと呼ぶ）に実装される認識エンジンがある．サーバ側で動作する認識エンジンは複数のユーザから利用される．それゆえ，想定外の規模の同時利用に動的に対応するため，スケーラブルな多重構成の実現が重要となる．一方，クライアント上で動作する認識エンジンは基本的には携帯端末のユーザのみが利用するため，特別な多重構成の仕組みは不要である．

そこで，図 2 に示すように，サーバ側に，シナリオ実装レイヤからの仮想認識エンジンへの認識実行要求と画像を登録するジョブキュー機能と画像キャッシュ機能を構成する．そして，シナリオ実装レイヤから認識実行要求と画像が登録される度に，認識実行要求をジョブキュー内の待ち行列に追加し，全アダプタに対して認識実行要求が追加された旨をブロードキャストする．各アダプタはジョブキュ

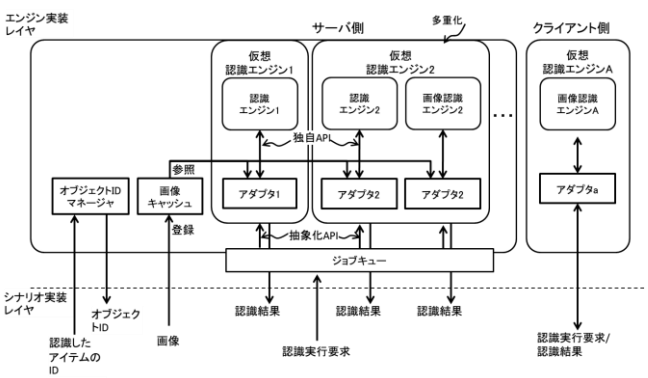

図 2　エンジン実装レイヤの構成要素

一内の自身への認識実行要求を取得し，画像キャッシュから対象画像を取得して認識処理を行い，認識結果をシナリオ実装レイヤに返す．

このような構成とすることで，新たな種類の認識エンジンの追加は専用のアダプタの開発とジョブキューへの登録のみで実現できる．また，新たな大量のリクエストの発生が見込まれる場合には，個々の認識エンジンの処理性能に応じて，認識エンジンとアダプタの稼働数を増やすだけで容易にスケールアウトすることが可能となる．

4.3 シナリオ実装レイヤ

本節では，サーバ上で動作する認識エンジンとクライアント上で動作する認識エンジンの統合方式を述べる．

シナリオ実装レイヤの構成を図 3 に示す．シナリオ実装レイヤには，サーバ上の仮想認識エンジンへの認識処理実行手順を記述するサーバサブシナリオ群，クライアント上の仮想認識エンジンへの認識処理実行手順を記述するクライアントサブシナリオ群，サーバサブシナリオ群とクライアントサブシナリオ群の実行手順を記述するシナリオドライバで構成する．

そして，シナリオドライバにはサーバサブシナリオとクライアントサブシナリオへのサブシナリオ実行要求とサブシナリオ実行結果を取得する基本機能を構成する．また，サーバサブシナリオとクライアントサブシナリオには，シナリオドライバからのサブシナリオ実行要求の取得とサブシナリオ実行結果の通知および仮想認識エンジンの抽象化 API を用いた認識実行要求と認識結果を取得する基本機能を構成する．

認識エンジンの統合ロジックは，シナリオドライバからの実行要求に応じた仮想認識エンジンへの認識処理の手順やオブジェクト ID マネージャに問い合わせながら認識結果に対する処理をクライアントサブシナリオとサーバサブシナリオに実装する．そして，クライアントサブシナリオとサーバサブシナリオの実行手順をシナリオドライバに実装する．

このように，サブシナリオでは，画像認識エンジンの API やアイテムの ID 体系の違い，多重化を考慮することなく，仮想認識エンジンへの実行要求と結果受付の組合せによって，画像認識エンジンの統合処理を記述する．また，シナリオドライバでクライアントサブシナリオとサーバサブシナリオを組み合わせることによって，サーバ上の認識エンジンとクライアント上の認識エンジンの統合処理が可能になる．

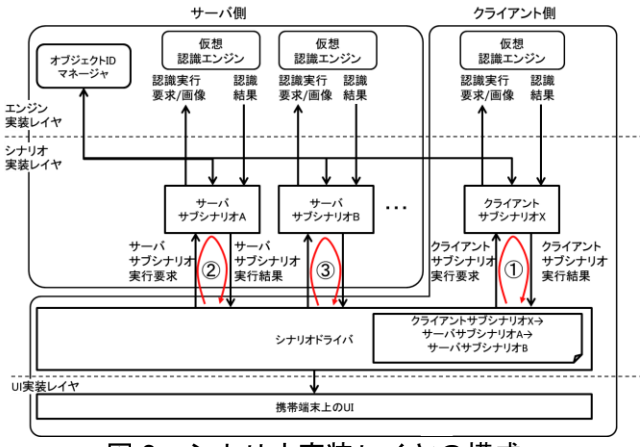

図 3　シナリオ実装レイヤの構成

5. 実証実験

本章では，NTT が主催するプライベート展示会で行った実験結果を述べる．実証実験では，UOR フレームワークに基づいて，展示パネル，展示オブジェ，デジタルサイネージ等のアイテムをスマートフォンのカメラで撮影することで，それらの詳細情報を画面表示する展示案内サービス（以降，「かざして案内」と呼ぶ）を開発した．

5.1 「かざして案内」概要

「かざして案内」は，展示会の公式アプリの 1 つの機能として実装した．「かざして案内」を利用している様子を図 4 に示し，画面遷移の例を図 5 に示す．

「かざして案内」のユーザインタフェースは，Android と iOS 向けのネイティブアプリで実現した．このアプリでは，展示会の来場者が認識したいアイテムに向けてスマートフォンのカメラをかざし，カメラプレビュー画面下部の「カメラボタン」をタップすると，認識されたアイテムに応じたダイアログを表示する．ついで，「詳しく見る」ボタンをタップすると認識結果の詳細情報表示画面を表示することとした．

「かざして案内」では，展示会の様々なアイテムを対象にすべく，物体認識エンジン，静止画透かしエンジン，動画透かしエンジンの 3 種類の認識エンジンを用いた．

認識対象のアイテムを図 6 に示す．また，個々のアイテムと認識エンジンとの対応を表 1 に示す．

「かざして案内」では，会場内の展示パネル，史料館の展示オブジェ，レストランの看板等の合計 153 点のアイテムを物体認識エンジンで認識することとした．また，レストランのテーブル 40 卓に特設した卓上ポップを，静止画透かしエンジンで認識することとした．なお，この卓上ポップには同じ絵柄の静止画を 2 つ並べてそれぞれに異なる透かし ID を埋め込み，認識結果として異なるコンテンツに誘導するようにした．さらに，展示動画に透かしを埋め込んだコンテンツを 1 カ所の展示ブースで再生し，動画透かしエンジンにて認識することとした．

5.2 システム構成

「かざして案内」を実現するためのシステム構成を図 7 に示す．サーバ上で動作する認識エンジンとして，アングルフリー物体検索技術[15]を実装した物体認識エンジンと画像認識型透かし埋め込み・検出技術[16]を実装した静止画透かしエンジンを用いた．また，クライアント上で動作する認識エンジンとして，MagicFinder [17]を動画透かしエンジンに用いた．

物体認識エンジンは 6 台のサーバに導入し，認識性能の向上を図った．これにより，物体認識エンジンは 6 多重とした．静止画透かしエンジンも 6 台のサーバに導入し，各サーバに 3 つの認識エンジンを配置した．これにより，静止画透かしエンジンは 18 多重とし，十分な処理性能を確保した．

エンジン実装レイヤのジョブキュー，画像キャッシュ，オブジェクト ID マネージャ，シナリオ実装レイヤのサーバサブシナリオは 1 台の UOR サーバに配置し，リバース

プロキシを介してスマートフォンからの実行要求を受け付ける構成とした．サーバ上のアダプタ，サーバサブシナリオ，オブジェクト ID マネージャは Node.js[18]で動作するアプリケーションとして実装した．ジョブキューは OSS のジョブキューサーバである Abraxas[19]をミドルウェアとして実装した．オブジェクト ID マネージャのデータベースおよび画像キャッシュは，OSS のキーバリューストアである Redis[20]をミドルウェアとして実装した．

クライアントのシナリオドライバ，クライアントサブシナリオ，アダプタはスマートフォンの OS 毎に実装した．具体的には Android 端末では Kotlin と Android Java，iOS 端末では Swift を実装言語とした．

図 4　「かざして案内」を利用している様子

図 5　「かざして案内」の画面遷移

図 6　認識対象のアイテム

表 1　アイテムと認識エンジン

アイテム	場所	種類	認識エンジン
展示パネル	展示会場	123	物体認識エンジン
展示オブジェ	史料館	22	
レストランの看板等	レストラン等	8	
卓上ポップ	レストラン	2	静止画透かしエンジン
展示動画（デジタルサイネージ）	展示会場内の特定ブース	2	動画透かしエンジン

5.3 シナリオドライバとサブシナリオの実装

「かざして案内」向けに実装したシナリオの処理を以下に述べる．物体認識エンジンや静止画透かしエンジンはスナップショットで取得した画像で認識が可能であるが，動画透かしエンジンは一定時間のフレーム画像を入力とする必要がある．そこで，カメラプレビュー画面を表示した時点から，シナリオドライバはカメラで撮影されたフレーム画像を入力としてクライアントサブシナリオを実行する．クライアントサブシナリオは動画透かしエンジンに対して認識要求を発行し，認識結果を一定時間保持する．そして，利用者がカメラボタンをタップしたとき，シナリオドライバは認識結果のキャッシュ有無を確認する．キャッシュがある場合，キャッシュされた認識結果をアプリケーションに返却する． キャッシュがない場合，カメラボタンをタップした時のフレーム画像を入力としてサーバサブシナリオを呼び出す． サーバサブシナリオは物体認識エンジン，静止画透かしエンジンに対してパラレルに認識要求を発行する． サーバサブシナリオは認識結果を受け付け次第，シナリオドライバに認識結果を返却する．

このように，実装した統合ロジックは，展示会の様々なアイテムを対象とすべく，サーバ上の認識エンジンとクライアント上の認識エンジンのパラレル処理を基本とした．また，動画透かしエンジンの処理を，カメラボタンをタップする前から動作させることで，ユーザビリティの向上に努めた．

5.4 実験結果と評価

展示会は 2017 年 2 月 13 日～17 日の 5 日間，10 時～17 時（13 日のみ 13 時～17 時）に開催され，約 1.2 万人が来場した．展示会の公式アプリの総ダウンロード数は 5433(iOS:3127, Android:2306)であった．公式アプリの利用ユーザのうち「かざして案内」を 1 度以上利用したユニークユーザ数は 1910 人であった．更に，「かざして案内」の画面でカメラボタンがタップされた回数，すなわち利用者が「かざして案内」で認識を試行した件数は 8005 件で

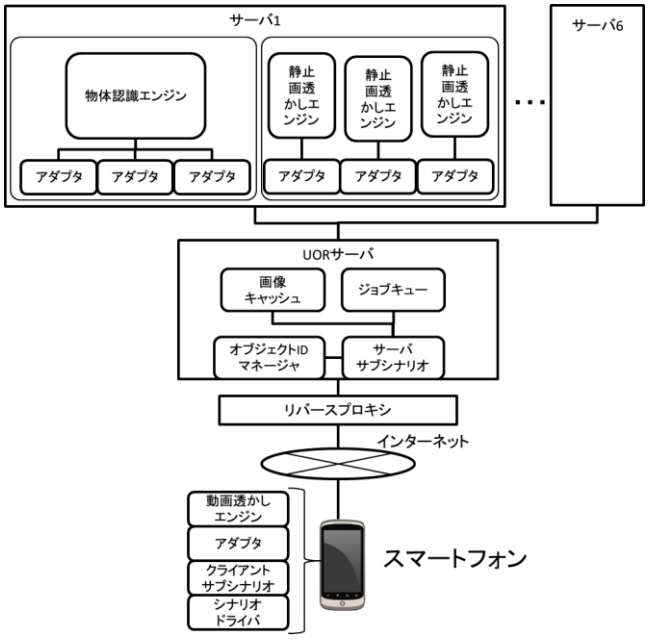

図 7　「かざして案内」のシステム構成

あった．また，1 時間あたりの試行が最も多かったのは，最終日(2/17)の 12 時～13 時の間で 377 件であった．

5.4.1 「かざして案内」の利用結果

展示会の期間中に，「かざして案内」で認識を試行した 8005 件のうち，いずれかの認識エンジンが認識結果を返したのは 5980 件であった．認識されたアイテムと認識件数の内訳を表 2 に示す．認識件数に違いはあるが，3 種類全ての認識エンジンでアイテムが認識されていることがいえる．認識件数は物体認識エンジンによるものが最も多い．中でも展示パネルが特に多いのは，パネルの種類が多くかつ，展示会場の各展示ブースで展示されており，利用者の接触機会が最も多かったためと考えられる．ついで，静止画透かしエンジンによるものが多い．静止画透かしエンジンの認識対象は卓上ポップのみだが，卓上ポップをレストランの各テーブル（40 卓）に設置しており，利用者が接触する機会は多かったことによると考えられる．動画透かしエンジンによる認識件数が最も少ないのは，展示動画を再生するデジタルサイネージが 1 つの展示ブースに限定され，利用者の接触機会が最も少なかったためと考える．

次に展示会最終日（2/17）の 1 時間あたりの認識件数の時系列推移を図 10 に示す．展示パネルは 10 時台，12 時台 16 時台と複数の時間帯で認識件数が増加する一方で，卓上ポップは 11 時から 12 時にかけて認識件数が最も多く発生している．卓上ポップはレストランに設置したため，レストランの来客数が増加する昼食時に認識件数も増加したと考えられる．これにより「かざして案内」が来場者の実際の動線に沿って利用されていたことが伺える．

以上より，5 日間に渡って，のべ 1910 人に利用して頂き，展示会の様々な場所で多様なアイテムを対象にした情報案内サービスを UOR フレームワークに基づいて実現できることを確認できた．

表 2　「かざして案内」の認識結果

アイテム	認識件数	認識エンジン
展示パネル	4493	物体認識エンジン
展示オブジェ	206	
レストランの看板等	971	
卓上ポップ	267	静止画透かしエンジン
展示動画（デジタルサイネージ）	43	動画透かしエンジン

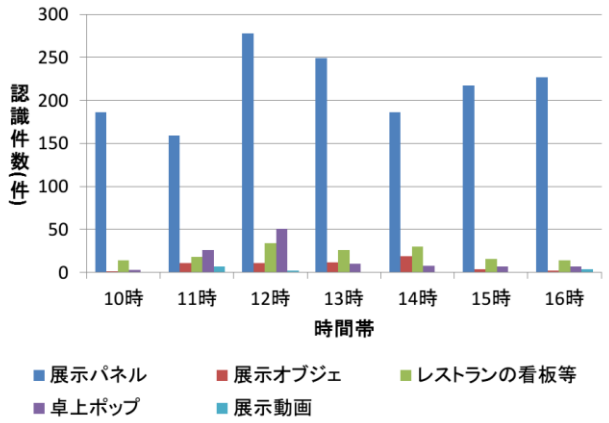

図 8　1 時間あたりの認識件数(2 月 17 日)

5.4.2 認識エンジンの多重化による効果

認識エンジンの多重化が有効に働いていれば，認識要求が増加してもサーバの処理時間は増加しないといえる．そこで，サーバ上の認識エンジンが認識結果を返した認識要求について，1 時間あたりの平均応答時間の時系列推移を確認した（図 9）．図 9 において，「認識件数」は，サーバ上の静止画透かしエンジンと物体認識エンジンによる 1 時間あたりの認識件数である．「サーバ」は，サーバにおける処理全体にかかった時間である．具体的にはサーバサブシナリオが実行要求を受けてから実行結果をシナリオドライバに返却するまでの時間であり，認識エンジンの処理時間を含む．「通信」は，クライアント（シナリオドライバ）とサーバサブシナリオの間の通信にかかった時間である．「全体」は「通信」と「サーバ」の合計であり，利用者がアイテムを撮影し，携帯端末が認識結果を受信するまでの時間である．

サーバの処理時間は，認識件数の増減に関わらず 1.4 秒から 2.0 秒の間を推移している．全体の応答時間と通信時間は認識件数が最も多い 2 月 17 日 12 時台に最も大きくなっているが，それ以外に認識件数の増減との目立った関連性は見られない．また，全体の応答時間と通信時間の推移

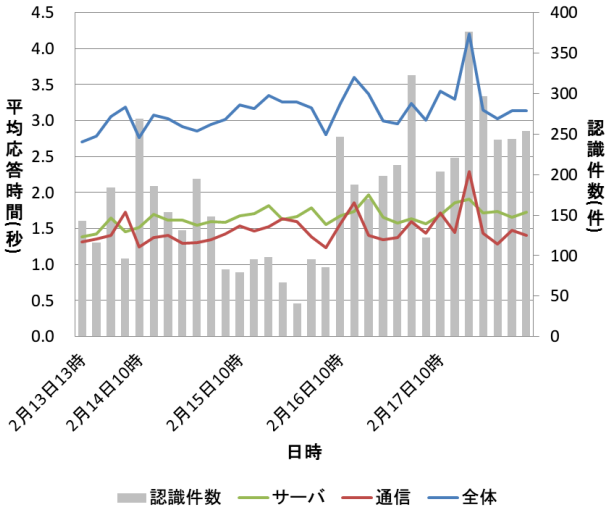

図 9　1 時間あたりの平均応答時間の推移

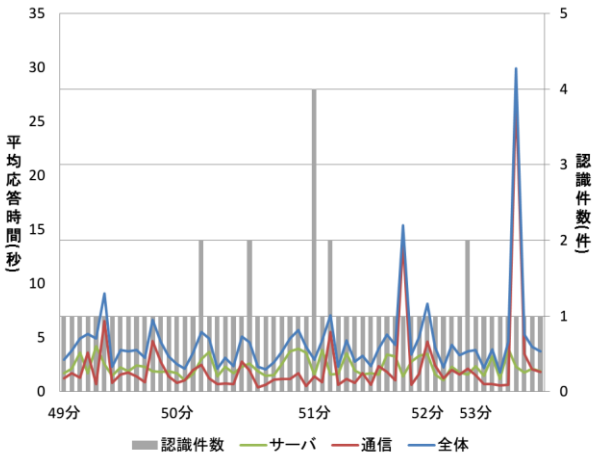

図 10　秒間認識件数ピーク前後における 1 秒あたりの平均応答時間の推移

は似通っており，通信時間の変動が全体の応答時間に影響していると考えられる．

さらに，サーバに最も負荷が集中した時間に着目し，1 秒あたりの平均応答時間のばらつきを確認した．サーバ上で 1 秒あたりの認識件数が最大になったのは 2 月 17 日 12 時 51 分 05 秒の 4 件／秒であった．上記時刻の前後 2 分間における 1 秒あたりの平均応答時間の時系列推移を図 10 に示す．

認識件数は 1 秒あたり 1 件～4 件の間で推移している．サーバの処理時間は，認識件数の増減によらず 1.0 秒から 4.2 秒の間を推移している．通信時間も認識件数によらず推移しているが，通信時間は時々大きく増加しており，全体の応答時間に影響を与えている．また，通信時間と全体の応答時間の推移は似通っており， 通信時間が全体の応答時間に影響していると考えられる．

以上より，実験中の処理負荷が最も集中する場合でも，認識件数の増加によってサーバの処理時間が増加している傾向は見られないことから，認識エンジンの多重化は有効に機能していると考える．

5.4.3 アプリケーション開発の効率性

UOR フレームワークによるアプリケーション開発効率性を評価する．ここでは実証実験システムにおいて，認識エンジンやシナリオの追加・変更によらず流用可能な機能と，新たに実装が必要な機能の開発規模の比率をみることで開発効率性を定量的に評価した．新たに実装が必要な機能の比率が小さいほど，新たな認識エンジンやシナリオを追加・修正する際の効率性が高いといえる．開発規模の集計は以下の条件で行った．

- 認識エンジンやシナリオの追加・変更によらず流用可能な機能を共通機能として集計する．具体的には，図 7 のジョブキュー，画像キャッシュ，オブジェクト ID マネージャ，アダプタにおける抽象化 API，シナリオドライバと各サブシナリオの基本機能を指す．
- 認識エンジンやシナリオを追加変更する場合に新たに実装が必要な機能を個別機能として集計する．具体的には，図 7 のアダプタにおける認識エンジンの独自 API，シナリオドライバと各サブシナリオで「かざして案内」向けに実装したシナリオを指す．
- 各認識エンジン，実証実験システムの実装に用いたミドルウェア，UI 実装レイヤに該当する「かざして案内」の UI は開発規模に含めない．

上記の条件に従って，開発したアプリケーションの共通機能と個別機能の SLOC(Source lines of code)を集計して割合を算出した．サーバとクライアントの Android 端末，iOS 端末それぞれで開発言語が異なるため，開発規模は別々に集計した．さらに Android 端末は共通機能と個別機能の開発言語が異なるため，どちらかの SLOC を補正する必要がある（共通機能：Kotlin，個別機能：Android Java）．Kotlin の開発元の Jetbrains 社によれば，Kotlin は Java のコード行数を概算で約 40%を削減するとされる[21]ことから，今回は個別機能の SLOC を 40%削減した上で，共通機能の SLOC との比率を算出した．

集計結果を表 3 に示す．個別機能の開発規模はサーバで 13%，クライアントで最大でも iOS の 46%と全体の半分以下であった．サーバがクライアントに比べて個別機能の比率が小さいのは，エンジン実装レイヤにおける多重化の機能と，オブジェクト ID マネージャをサーバのみで提供しているためと考える．

さらに，個別機能を各認識エンジンのアダプタ，各サブシナリオ，シナリオドライバに細分化し開発規模の比率を算出した．サーバの開発規模の比率を表 4 に，クライアント（iOS）の開発規模の比率を表 5 に示す．

個々のアダプタが開発規模に占める比率は，サーバで 10%未満，クライアントで 20%未満であった．これにより，認識エンジンを入れ替える場合，修正規模はシステム全体の 10～20%程度に抑えられることが期待できる．シナリオドライバと各サブシナリオが開発規模に占める比率は，クライアントは 29%，サーバは 1%であった．これにより，シナリオの追加変更に伴う修正範囲はシステム全体の 30%程度に抑えられることが期待できる．

以上より，UOR フレームワークの設計指針に基づくことで，実証実験相当のシステム開発で 50%以上の開発規模の効率化が期待できること，また，認識エンジンの入れ替えや統合ロジックの追加変更に対して，システム全体の 10～30%程度の開発規模に抑えて実現することが期待できることを確認した．

6. まとめ

様々なアイテムを対象とした携帯端末向け情報案内サービスの実現を狙い，(1)認識エンジン，統合ロジック，UI の実装レイヤの分離により認識エンジンの追加や統合を効率化する基本構造，(2)アダプタとジョブキューにより認識エンジンの API の抽象化とスケーラビリティの確保を実現するエンジン実装レイヤ，(3)サブシナリオとシナリオドライバにより，サーバとクライアントの認識エンジンの統合処理を簡易に記述可能とするシナリオ実装レイヤを特徴とした UOR フレームワークを提案した．実証実験では 1 万人規模の展示会での展示案内サービスを UOR フレームワークにより具現化し，以下を確認した．

- クライアントとサーバにまたがる複数の認識エンジンの

表 3　実証実験システムにおける共通機能と個別機能の開発規模比率

集計対象	開発規模の比率	
	共通機能	個別機能
サーバ	87%	13%
クライアント（Android）	63%	37%
クライアント（iOS）	54%	46%

表 4　サーバの開発規模比率

機能分類		比率
共通機能		87%
個別機能	アダプタ（物体認識エンジン）	9%
	アダプタ（静止画透かしエンジン）	3%
	サーバサブシナリオ	1%

表 5　クライアント（iOS）の開発規模比率

機能分類		比率
共通機能		54%
個別機能	アダプタ（動画透かしエンジン）	17%
	クライアントサブシナリオ	6%
	シナリオドライバ	23%

統合が可能であること
- サーバ上の認識エンジンの多重化により安定した処理性能の確保が可能であること
- 同種のシステムを新規開発する場合に比べて 50%以上の開発規模の効率化が期待できること

今後は，統合した認識エンジンの高度な利用方針の検討や，さらなるアイテム数の大規模化への対応等を進める．

参考文献

[1] B. Girod, V. Chandrasekhar, D. M. Chen, N. Cheung, R. Grzeszczuk,Y. Reznik, G. Takacs, S. S. Tsai, and R. Vedantham, “Mobile Visual Search”, IEEE Signal Processing Magazine, Vol. 28, Issue 4 (2011).

[2] J. J. Hull, B. Erol, J. Graham, Q. Ke, H. Kishi, J. Moraleda and D. G. Van Olst, “Paper-Based Augmented Reality”, In Proc. of IEEE 17th Int. Conf. on Artificial Reality and Telexistence, pp.205-209 (2007).

[3] S. S. Tsai, D. Chen, V. Chandrasekhar, G. Takacs, N. Cheung, R. Vedantham, R. Grzeszczuk and B. Girod, “Mobile product recognition”, Proc. of the 18th ACM int. conf. on Multimedia, pp.1587-1590 (2010).

[4] B. Erol, E. Antúnez and J. J. Hull, “HOTPAPER: Multimedia Interaction with Paper using Mobile Phones”, In Proc. of the 16th ACM int. conf. on Multimedia, pp.399-408, 2008).

[5] X. Yang, Y. Tian, C. Yi and A. Arditi, “Context-based indoor object detection as an aid to blind persons accessing unfamiliar environments”, Proc. of the 18th ACM int. conf. on Multimedia, pp.1087-1090(2010).

[6] 増田, 宮越, “被写体認識技術＋OCR 技術を活用したスマートフォンアプリによるマイナンバー収集代行サービスの実用化”, 情報処理学会デジタルプラクティス, Vol.8, No.1, pp.67-72 (2017).

[7] S. Deshpande and R. Shriram,“Real time text detection and recognition on hand held objects to assist blind people”, Proc. of 2016 Int. Conf. on Automatic Control and Dynamic Optimization Techniques (ICACDOT), pp.1020-1024(2016).

[8] S. S. Tsai, H. Chen, D. Chen, G. Schroth, R. Grzeszczuk and B. Girod, “MOBILE VISUAL SEARCH ON PRINTED DOCUMENTS USING TEXT AND LOW BIT-RATE FEATURES”, In Proc. of 2011 18th IEEE Int. Conf. on Image Processing (ICIP), pp.2061-2604 (2011).

[9] S. S. Tsai, D. Chen, H. Chen, C. Hsu, K. Kim, J. P. Singh and B. Girod,“Combining image and text features: a hybrid approach to mobile book spine recognition”, Proc. of the 19th ACM int. conf. on Multimedia, pp.1029-1032 (2011).

[10] K. B. Raja. R. Raghavendra, M. Stokkenes and C. Busch,“Multi-modal Authentication System for Smartphones Using Face, Iris and Periocular”, Proc. of Int. Conf. on Biometrics(ICB), pp.143-150 (2015).

[11] S. Gammeter, A. Gassmann, L. Bossard, T. Quack and L. V. Gool,“Server-side object recognition and client-side object tracking for mobile augmented reality”, Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, pp.1-8 (2010).

[12] Google, CLOUD VISION API, https://cloud.google.com/vision/?hl=ja(2017 年 6 月 20 日閲覧)

[13] Computer Vision API, https://azure.microsoft.com/ja-jp/services/cognitive-services/computer-vision/(2017 年 6 月 20 日閲覧)

[14] Amazon, Amazon Rekognition, https://aws.amazon.com/jp/rekognition/(2017 年 6 月 20 日閲覧)

[15] J. Shimamura, T. Yoshida, and Y. Taniguchi, “View-Directional Consistency Constraints for Robust 3D Object Recognition”, IIEEJ trans. image electronics and visual computing, Vol.3, No.2, pp.164-173 (2015).

[16] 安藤, 五十嵐, 杵渕, 中村, 並河, 山下, 八尾, 草地, 武井, “コンビニ向け商品情報提示サービスを実現する画像認識型透かし埋め込み・検出技術”, NTT 技術ジャーナル, Vol.29, No.6, pp.10-15 (2017).

[17] NTT テクノクロス, MagicFinder 動画や静止画とスマートフォンの連携を実現，https://www.ntt-tx.co.jp/products/magicfinder/(2017 年 6 月 20 日閲覧)

[18] Node.js Foundation, Node.js, https://nodejs.org/ja/(2017 年 6 月 20 日閲覧)

[19] R. Turner, Abraxas, https://github.com/iarna/abraxas(2017 年 6 月 20 日閲覧)

[20] Redis Labs, Redis, https://redis.io/(2017 年 6 月 20 日閲覧)

[21] JetBrains, Kotlin Programming Language, https://kotlinlang.org/ (2017 年 6 月 20 日閲覧)