

卒業研究報告

題目

画像情報によるスマートセルフ精算システムの開発

指導教官

高橋寛教授

王森レイ講師

報告者

段原丞治

令和～～年～月～日提出

愛媛大学工学部情報工学科情報システム工学講座

目次

第 1 章 まえがき	1
第 2 章 準備	3
2.1 設計に用いた用語解説	3
2.2 画像処理に用いた技術の用語解説	5
第 3 章 システムの設計	6
3.1 システムの概要	6
3.2 要求定義	7
3.3 設計	9
第 4 章 実装内容・検証結果	16
4.1 実装・実験環境	16
4.2 検証結果	23
4.2.1. 単体テスト	23
4.2.2. 結合テスト	25
4.2.3. 総合テスト	26
第 5 章 システムの評価・考察	27
第 6 章 まとめ	29
謝辞	30

参考文献

31

第 1 章

まえがき

現在の日本では少子高齢化の進行による人的資源が減少している。総務省の平成 28 年度の人口調査では 2030 年には 1 億 1662 万人に減少し、2050 年には人口が 1 億人下回ると見込まれる [1]。人的資源の減少という問題は社会全体に関わることである。人手不足問題解消のために、各産業では業務の無人化が急務となっている。

最近、サービス業者にも人手不足の問題が深刻化おり、セルフレジの導入が進んでいる。しかしながら、セルフレジは導入コストが高いという問題がある。セルフレジは、登録機と精算機を合わせて平均で 300 万を超える [2]。セルフレジの導入は、人手不足問題を抱える店舗においてメリットも大きい負担も大きい。特に小規模な店舗の導入における負担が大きく、既存のセルフレジより安価で導入しやすいシステムが求められている。

これらの問題を解決するために、資金力を持たない店舗でも導入しやすく、安価で人手のかからないシステムの提案と開発を本研究の目的とした。本研究の目的としては、バーコードスキャン技術を利用したスマートモビリティレジシステムを開発することである。

本研究では、商品の識別から決算までのシステム構築を V 字開発モデルに従い、グループ (段原丞治、真鍋樹) で研究を行った。要求定義や設計は UML(Unified Modeling Language) 図を用いて定義し、それをもとにシステムの設計と実装を行う。実装では、エッジ処理側とサーバ処理側で担当を分けた。エッジ側のハードウェアは RaspberryPi を使用した。サーバは画像処理を Yolo(画像から物体を識別する機械学習ネットワー

ク)[5] や OpenCV(画像処理ライブラリ)[7] 等を利用して行う。

本論文は、以下のような構成をとる。第 2 章では、本研究で使用した用語、技術の解説を述べる。第 3 章では、UML 図を用いてシステムの要求定義、設計、検証項目を述べる。第 4 章では、実装内容と検証結果を示す。第 5 章では、実装したシステムの評価及び考察を述べる。第 6 章では、本研究のまとめを示す。

第 2 章

準備

本章では本研究で使用した用語、技術について述べる。本研究は V 字開発モデルに従って開発を行い、設計に関しては UML 言語を用いた。実装部分の画像処理では Yolo と pyzbar を使用した。以下にそれぞれの用語の定義と、本研究との関連も込めて述べる。V 字モデルと設計に関する用語は 2.1 節で述べ、画像処理に関わる用語と技術の説明は 2.2 節に述べる。

2.1 設計に用いた用語解説 (参考文献)

V 字開発モデル

V 字開発モデルとはシステム開発の要求分析、基本設計、詳細設計、実装に分けて時間軸順に検証、テストを行う開発プロセスである。細かい要素ごとにテストをしていくことで、開発の途中で大幅な変更や問題が起きにくくなる利点がある。本研究では V 字モデルに従って、開発を行っている。

UML(Unified Modeling Language)

UML とはオブジェクト指向のソフトウェア開発において、データ構造や処理の流れなどソフトウェアに関連する様々な設計や仕様を図面で可視化するためのモデリング言語である。UML の利点は、単純で分かりやすい点と、実用的な表現方法を持つ点で

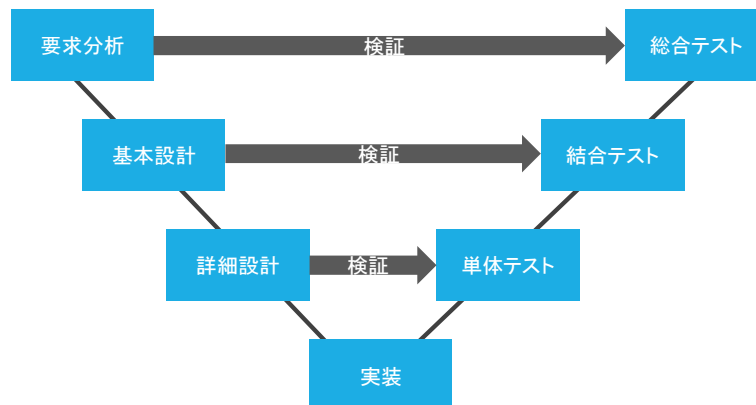


図 2.1. V字モデル

ある。UML の構成要素としてユースケース図、クラス図、シーケンス図がある [4]。本研究ではシステムの設計を行う際に、曖昧な定義になるのを防ぐために UML 図を用いた。

ユースケース図

ユースケース図とは、UML で定義されている図のうちのひとつであり、システムがどのように機能すべきかという振る舞いとその外部環境を表す。ユースケース図を作成することで視覚的にわかりやすくなる。ユースケース図を用いて、開発者だけでなく、クライアントにもシステムの概要が理解しやすくなる。本研究では、基本設計を行う際に使用した。

シーケンス図

シーケンス図とは、UML で定義されている相互作用図の一種類である。システムを構成する、オブジェクト間のメッセージのやり取りを時系列順に沿って並べて表現したものである。四角の項目はオブジェクトと呼ぶ。矢印はメッセージと呼ばれ、意味はオブジェクト間のデータの伝達と動作が行われる。ユースケース図を基にしてシーケンス図やクラス図を制作を行う。本研究では、詳細設計を行う際に使用した。

クラス図

クラス図は UML の基本となる図のひとつである。システムの静的な構造を示す図である。クラスが持つ動作や、属性を記述する。より具体的な実装部分に近い記述を行う。本研究では、クラス図を参考にしながらシステムのモジュールを実装した。

2.2 画像処理に用いた技術の用語解説

Yolo(Real-Time Object Detection)

Yolo はリアルタイムでのオブジェクト識別が可能ネットワークである。Web カメラを利用したリアルタイム検出も可能である。ほかのネットワークとの違いは、検出と識別を同時に行っているため高速性を保てる点である [5]。本研究では、画像からバーコードの位置を特定し、バーコード部分のみを切り取るために使用した。

pyzbar

pyzbar とは、python 言語でバーコード画像を解析し、数字を識別するライブラリである [6]。1 次元バーコード以外にも、2 次元バーコードである QR コードの読み取りも行うことが可能。本研究では、バーコード画像からバーコード番号を識別するために使用した。

第 3 章

開発するシステムの設計

本章ではスマートモビリティレジシステムの概要と、V 字開発モデルによる要求定義、設計および検証項目について述べる。本章の構築は以下の通りである。3.1 節では商品識別システムの概要を図で説明する。3.2 節ではユースケース図を用いて要求定義を説明する。3.3 節では、クラス図を用いて、基本設計及び詳細設計を述べる。さらに、クラス図に示されている各モジュールに対して検証項目を設定した。

3.1 システムの概要

スマートモビリティレジシステムでは、エッジデバイスとなる買い物カゴに RaspberryPi と各種センサを設置する。エッジ側では商品情報（商品名、金額、賞味期限、製造日など）の特定に必要となるデータを、センサ類を用いて取得する。データの取得後サーバ側にデータを送信し、サーバ側で画像データの処理を行い、商品を特定する。システムの流れを以下の図 3.1 に示す。

図 3.1 の左側の RaspberryPi で商品に関するデータをサーバに送信する。フラグは、送信された画像の商品をカゴへ追加するか、削除するかどちらかを判断するために使用する。サーバは画像データとフラグを受信する。画像からバーコード番号が識別できた場合、追加・削除のフラグに従って DB を更新する。そして、RaspberryPi に識別の結果を返信する。

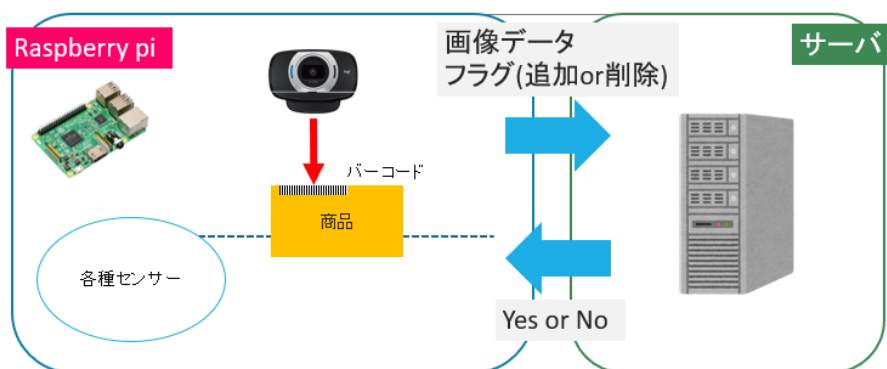


図 3.1. システムの流れ

3.2 要求定義

ユーザがスマートモビリティレジシステムに求める機能と動作をユースケース図を用いて表す。3.2 には、ユースケース図を示す。そこで、赤枠で囲っている部分が筆者が担当する。ユースケース図では、ユーザがスマートモビリティレジシステムに対して動作を行った際に、システムがどのような振る舞いをするかを示す。ユーザは通常の買い物のように、カートに商品を出し入れすることができる。このときカート (RaspberryPi) が商品に関するデータを集める。解析システムでは、商品情報の特定とカゴ DB への操作を行う。解析システムはサーバで動作する。カゴ DB は、カート内にある商品情報を管理し、最終的な決済システムで利用される。ユーザがカートを返却すると、決済システムが動作する。決済システムは、顧客の口座 (プリペイドカードや電子マネー口座) からカート内の商品合計金額を引く処理を行う。

次に、ユースケース図を基にスマートモビリティレジシステムのシナリオを以下の表 3.1 に示す。

表 3.1. スマートモビリティレジシステムのシナリオ

	シナリオ
買い物	商品を置く→バーコード認識→商品 DB 追加・削除→結果通知
決済	ユーザが退店する→決済を行う

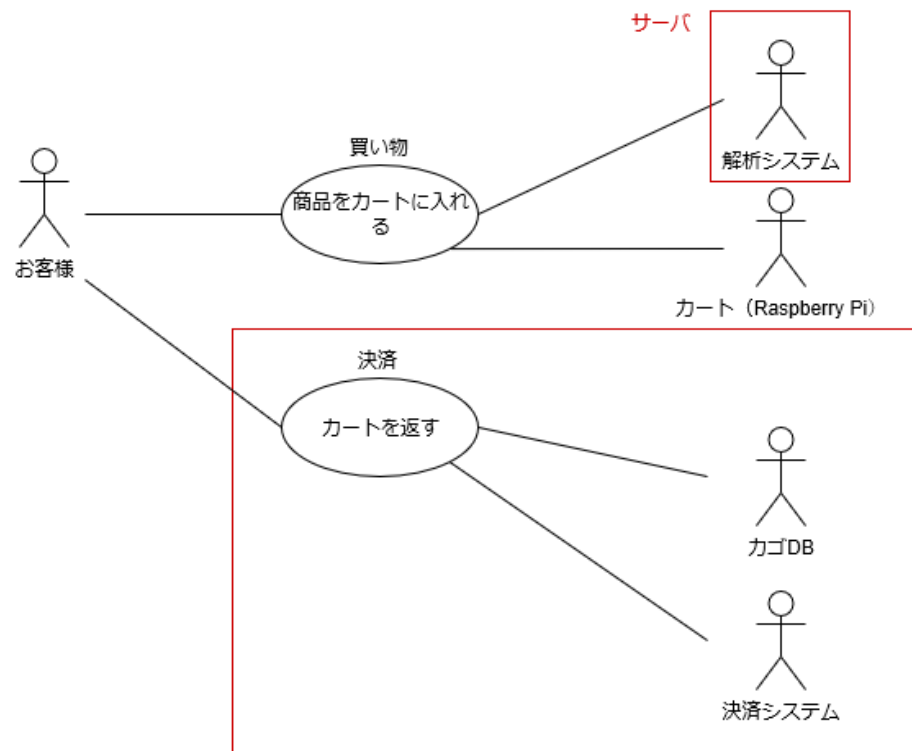


図 3.2. ユースケース図

次に、要求定義を満たす機能テスト項目を以下の表 3.2 を示す。

表 3.2. 総合テスト項目

シナリオ	確認の流れ
買い物	1-1 商品をカゴに入れ、超音波センサ反応時画像を撮りLEDを点灯させる。
	1-2 重量センサより商品追加・削除の情報をサーバへ送る。
	1-3 バイナリーデータを画像にフラグに変換する。
	1-4 Yoloを使用して画像の中のバーコードのみを切り取る。
	1-5 バーコード番号を解析する。
	1-6 商品のバーコード情報を確認する。
	1-7 商品を商品DBに追加・削除する。
	1-8 商品を正常に商品DBに追加できたかどうか結果をLED点灯にて通知する。
決済	2-1 カートを選択する。
	2-2 商品を確認し、決済を行う。

3.3 設計

基本設計

ユースケース図の動作を満たすように基本設計を行う。図 3.3 を基に、基本設計を行った。

図 3.3 に示されている項目をもとに、機能別にクラス化を行った。クラスには、カゴ、解析システム、カゴ DB、決済システム、商品 DB がある。クラス名カゴの役割は、ユーザがカート内に商品を出し入れする動作全般を行う。クラス名解析システムの役割は、カートに入れられた商品の識別を行う。カゴ DB の役割は、カートに入れられた商品の管理を行うためにある。決済システムの役割は、ユーザが決済を行う際の購入金額の計算および、カゴ DB の整理を行う。最後の商品 DB の役割は、商品の値段や、商品名、バーコード番号の管理を行うためにある。

図 3.3 はカゴ、解析システム、カゴ DB、商品 DB、決算システム、それぞれの操作を示したクラス図である。カゴクラスでは、ユーザが商品を出し入れする際に、その商品の画像データを確保する。また、ユーザに解析が成功したかの通知を行う。解析システムクラスでは、カゴから送られてきたデータの受信、解析とカゴ DB への操作

を行う。カゴ DB では、ユーザが購入予定の商品の管理を行う。商品 DB では、商品のバーコード番号と商品名、値などの商品情報を管理している。店側が新しい商品を追加する場合は、この DB に商品情報を追加する必要がある。最後の決済システムでは、ユーザが決済する際に必要な処理を行う。

次に、クラスごとの動作要件を満たす検証項目を表 3.3 に示す。

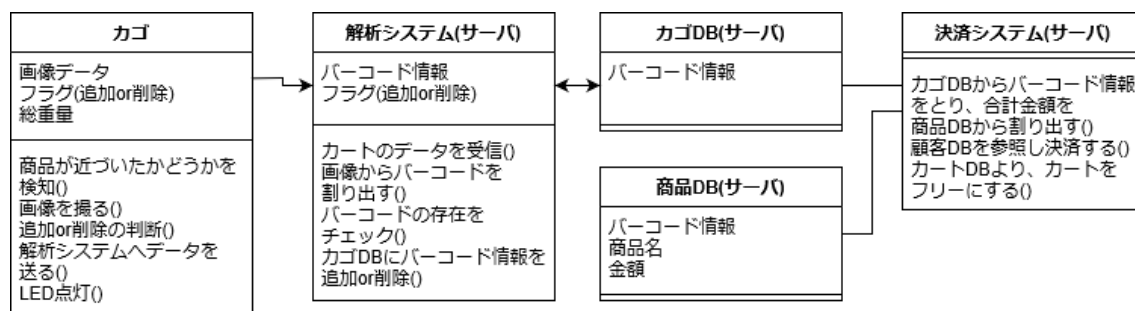


図 3.3. クラス図

詳細設計

次に、シーケンス図を用いてスマートモビリティレジシステムの手順を説明する。以下の図 3.4 にそって説明する。赤枠で囲っている部分が筆者の実装担当である。図 3.4 はカート側、サーバ側、Web ページ側の 3 つに大別される。左から右の順に処理を行う。シーケンス図から読み取れるように、ユーザからカートへ商品が渡され、カートで処理を行う。処理が終わると解析システムへ情報が伝達され、解析が始まる。解析結果をカゴ DB とカートに送信する。結果は Web ページへ反映される。カートでは DB の操作は行われない。

図 3.4 において筆者が担当した部分はメッセージの 7～10、12～18 である。図 3.4 に記述されている筆者が担当したメッセージの具体的な機能と、検証項目 (単体テスト項目) について説明していく。

7. サーバが、RaspberryPi から送信された画像データとフラグを受信する。
8. 受信した画像データを画像に変換する。画像を解析してバーコード番号を識別する。

		確認内容
サーバ側	受信	ラズパイからTCPで送られていた画像を正常に受信
		ラズパイからTCPで送られていた画像とフラグ(追加or削除)を正常に受信
		データ受信プログラムで受け取った画像データをYoloに渡し結果を取得
		Yoloから受け取った座標値をもとにバーコードのみを切り取る
		pyzbarにバーコード画像を渡して番号を取得
		バーコード番号をカートDBに追加
		エッジ側に画像受信から番号取得までが成否を返す
		商品を置いたとき、商品の重量を取得する
	Yolo	Windows10でYoloがPythonで実行できる
		任意の学習データを学習させバーコード専用の識別機を作る
		デフォルトのYoloのpythonプログラムをクラスに変更する
		OpenCV形式の画像をyoloに投げたらオブジェクト名と座標がついたリストが返ってくる
	バーコード判別	バーコード画像を番号に変換できる
	DB	カートごとに画面を分かれて表示させる
		カートの中にある商品の情報をリスト表示させる
		カート内にある商品で同じものがあれば個数を表示させる
		決算するとカート内にある商品がカート内DBから消える
		決算すると購入者である顧客の所持金が正常に変わる
		決算完了画面に残高、購入額を表示させる
		決算時に顧客の所持金が足りていなければ決算されないようにする
		商品が登録されているかの結果が返ってくる
		フラグから商品の追加、削除を判断してDBに反映する
		DBにある画像URLからWeb上に画像を表示させる

表 3.3. 結合テスト項目

9. 受信したフラグを基に、バーコード番号をカゴ DB に追加・削除する。
10. サーバから RaspberryPi へ、解析が成功したかどうかの結果 (フラグ) を返信する。
12. 購入予定商品を管理しているカゴ DB を参照し、結果を Web ページに表示する。Web ページ上では、購入予定商品が、リスト上に表示される。
13. ユーザは Web ページで購入予定商品を確認し、購入ボタンを押すことで購入が決定する。
14. 決済システムがカゴ DB に接続し、決済を行う。
15. 商品 DB を参照して、ユーザが購入する商品の合計金額を割り出す。なお、商品 DB を参照する理由は、カゴ DB には購入予定商品のバーコード番号しか管理されているためである。
16. メッセージ 15 で割り出した合計金額を Web ページに表示する。
17. 顧客 DB に登録されているユーザの所持金から、合計金額を引き決済する。なお、所持金が購入金額より下回っていた場合は、決済は行われない。
18. 決済が終了すると、Web 画面にユーザの残高が表示される。

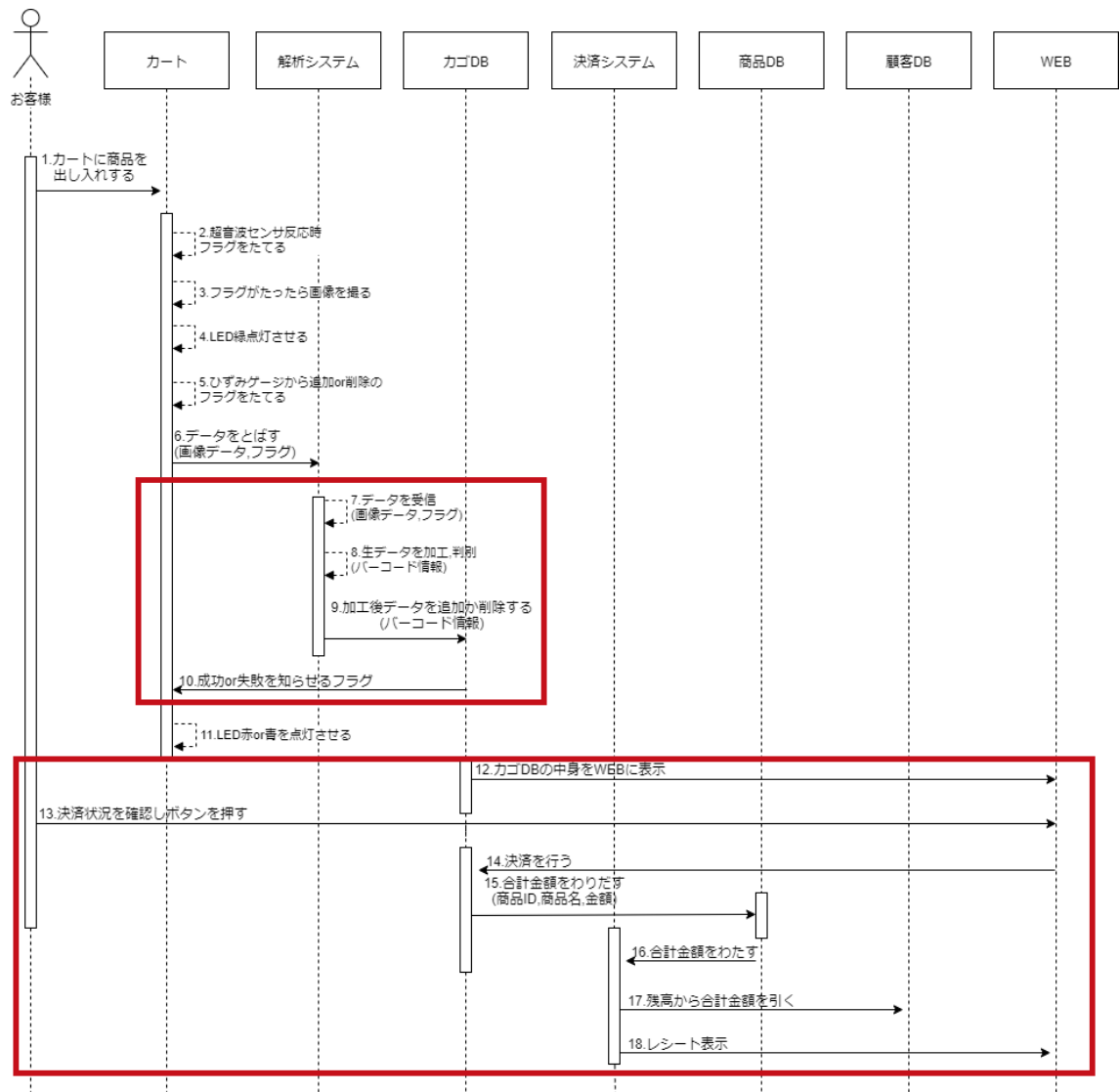


図 3.4. シーケンス図

次に、筆者が担当した部分が、詳細設計の要件を満たすかの検証を行う。単体テスト項目を作り、検証を行った。

表 3.7 は、メッセージ番号の 7,9,10 番の RaspberryPi からのデータ通信に関する単体テスト項目である。

表 3.4. サーバ単体テスト項目

テスト	確認項目
1	RaspberryPiからTCP通信で送られてきた複数の画像を正常に受信
2	RaspberryPiからTCPで送られてきたフラグ(追加or削除)を正常に受信
3	データ受信後、受け取った画像データをYoloに渡し結果を取得
4	Yoloから受け取った座標値をもとにバーコード領域のみを切り取る
5	pyzbarにバーコード画像を渡して番号を取得
6	バーコード番号をカートDBに追加
7	RaspberryPiに解析成功の成否をTCP通信で返す

表 3.5 は、メッセージ番号 8 のバーコード解析についての単体テスト項目である。

表 3.5. Yolo 単体テスト項目

テスト	確認事項
1	YoloをWindows10環境のGPUで動作できるようにコンパイル
2	任意の学習データを学習させバーコード専用の識別機を作る
3	Yoloをpythonプログラムで実行できるように修正
4	OpenCV形式の画像をyoloに投げたらオブジェクト名と座標が ついにになったりリストが返ってくる
5	カメラから15cmの距離でもYoloが検知率が80%以上を達成

表 3.6 はメッセージ番号 12～18 の決済システムに対する単体テスト項目である。

表 3.6. 決済システム単体テスト項目

テスト	確認項目
1	カートごとに画面を分かれて表示させる
2	カートの中にある商品の情報をWebページにリスト表示させる
3	カート内にある商品で同じものがあれば個数を表示させる
4	決済するとカート内にある商品がカート内DBから消える
5	決済すると購入者である顧客の所持金が正常に変わる
6	決済完了画面に残高、購入額を表示させる
7	決済時に顧客の所持金が足りていなければ決算されないようにする
8	商品が登録されているかの結果が返ってくる
9	フラグから商品の追加、削除を判断してDBに反映する
10	DBにある画像URLからWeb上に画像を表示させる

表 3.4 は、メッセージ番号 12 の商品 DB に対する単体テスト項目である。

表 3.7. 商品 DB 単体テスト項目

テスト	確認項目
1	商品名、値段、バーコード番号、サムネイルURLの登録ができる
2	商品名、値段、バーコード番号、サムネイルURLの削除ができる

第 4 章

実装内容・検証結果

本章では V 字開発モデルにおける実装及び検証について述べる。ここでは具体的な実装方法について述べる。

4.1 実装・実験環境

実装はグループで行い、エッジ処理側は真鍋樹が行った。この章では筆者が担当した、サーバ側の実装についてのみ述べる。ここでは、エッジ処理側のことをクライアントと呼ぶ。実装に使用したプログラム言語は、クライアント、サーバのどちらも Python3 である。

実験環境

実験環境で使用したハードウェアのスペックを以下の表 4.1 に示す。

表 4.1. 実行環境

処理担当	OS	CPU	RAM	GPU
サーバ側	Windows10 64bit Pro	Core2Duo E7500 2.93GHz	4GB	GT740
エッジ側	RaspberryPi 3B(Rasbian)	Broadcom 1.2GHz	1GB	None

実験環境で使用したライブラリ、プログラム言語を以下の表 4.2 に示す。

表 4.2. 使用ライブラリ・使用言語

処理担当	ライブラリ・使用言語名	バージョン	使用目的
サーバ側	Python3	3.7.4	メイン処理
サーバ側	OpenCV	4.1.2	画像切り取り
サーバ側	Yolo	3	バーコード領域取得
サーバ側	CUDA	10.1	学習に使用
サーバ側	pyzbar	0.1.8	バーコード番号取得
サーバ側	mysqlclient	1.4.6	DB 操作
Web ページ	XAMPP	3.2.4	Web ページ、DB ホスト
Web ページ	apache2	2.4.41	Web ページホスト
Web ページ	MariaDB	10.4.10	DB
Web ページ	PHP	7.3.12	Web ページ処理、DB 操作
エッジ側	Python3	3.7.3	メイン処理
エッジ側	OpenCV	3.4.3	Web カメラ操作

サーバ通信

クライアントとのデータのやり取りを含めた連携には、通信処理が必要不可欠になる。クライアントは、距離センサの反応後、カメラを起動し複数の画像を撮影する。クライアントからサーバへ送信するデータは、画像データとフラグをセットしたものである。しかしながら、ソケット通信において送信処理が複数回であったとしても、受信処理ではひとまとまりのデータとして受け取られることがある。その問題を防ぐために、送信データのヘッダにデータのサイズを書き込む手段をとった。クライアントから送られてきた画像データはバイナリ形式になっているので、OpenCV[7] のフォーマットに変換する。

Yolo によるバーコード領域特定

Yolo を使用した理由は、pyzbar[6] の識別精度にある問題があったからである。pyzbar[6] は、近距離で撮影したバーコードの画像の認識はできるが、距離が離れると識別しなくなる。これは、画像の中に占めるバーコード部分が少なくなることが原因であると考えられる。そこで、Yolo を使用し画像からバーコードの部分の座標を取得する。画像のうち、バーコードの部分のみを pyzbar に渡すことで、距離が離れていても近距離で撮影したのと同じ効果が得られるようになった。学習には、バーコードの画像を約 2000 枚用意した。Yolo の実行はサーバ側で行う。当初、クライアント側である RaspberryPi で Yolo を実行すればサーバは不要になり、通信におけるタイムラグもなくなることが仮定された。ところが、RaspberryPi の性能では Yolo を、高い識別精度を保ったままリアルタイムに動作させることは、性能上難しいと判断したためサーバで行うことになった。

DB を使用した商品情報の管理


カゴの中の商品の管理と、商品自体の情報の管理のために、DB を使用した。本研究では MariaDB を使用した。カゴ DB の構造を以下に示す。始めに、商品自体のデータを管理する item テーブルを表 4.3 に示す。jan とは JAN コードのことであり、スマートフォンリレジ番号である。title は、商品名のことである。price は、商品の価格を示す。image_url は、商品の画像がある URL を示している。最後の、image_raw はこのシステムでは使用していないが、画像データそのものを格納する。

表 4.3. item テーブル

	#	名前	データ型	照合順序	属性	NULL	デフォルト値	コメント	その他
<input type="checkbox"/>	1	jan	bigint(20)			いいえ	なし		
<input type="checkbox"/>	2	title	text	utf8mb4_general_ci		いいえ			
<input type="checkbox"/>	3	price	int(11)			いいえ	なし		
<input type="checkbox"/>	4	image_url	text	utf8mb4_general_ci		はい			
<input type="checkbox"/>	5	image_raw	blob			はい	NULL		
<input type="checkbox"/>	6	id 	int(11)			いいえ	なし		AUTO_INCREMENT

次に、cart テーブルの構造を表 4.4 に示す。id は、テーブルのレコードの固有番号を示すためにある。jan は、JAN コードのことである。cart_id はカートの固有番号を示す。この番号でカートごとの区別を行う。この番号があることで複数カートを運用した際も区別することができる。

表 4.4. cart テーブル

	#	名前	データ型	照合順序	属性	NULL	デフォルト値	コメント	その他
<input type="checkbox"/>	1	id 	int(11)			いいえ	なし		AUTO_INCREMENT
<input type="checkbox"/>	2	jan	bigint(20)			いいえ	なし		
<input type="checkbox"/>	3	cart_id	int(11)			いいえ	なし		

customer テーブルの構造を表 4.5 に示す。この customer テーブルは、顧客情報を管理する。id は顧客の固有番号を示す。balance は、顧客の所持金額を示す。

表 4.5. customer テーブル

	#	名前	データ型	照合順序	属性	NULL	デフォルト値	コメント	その他
<input type="checkbox"/>	1	id 	int(11)			いいえ	なし		AUTO_INCREMENT
<input type="checkbox"/>	2	balance	int(11)			いいえ	なし		

決済システム

設計段階では、決済システムの動作は、ユーザが退店する際に自動で行われる。しかし、その機能を実装するには時間の都合上難しいと判断したため、仮の決済用の Web ページを作成してテストを行っている。Web ページ作成には PHP と Html を使用した。以下に Web ページの動作の手順を示す。

カート画面

以下の図 4.1 は、どのカートを使用するか選択するためにある。

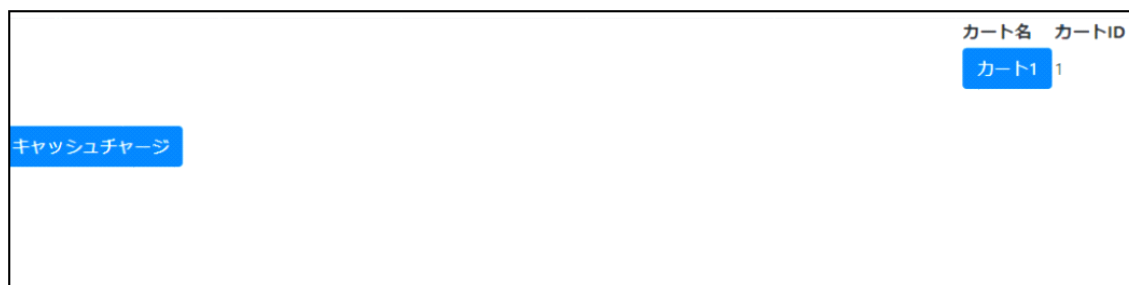


図 4.1. Web ページ カート画面

カート内商品一覧

図 4.2 は、カート内にある購入予定商品の一覧を表している。表示内容は、商品名と価格、商品画像、個数の 4 つである。ページ内にある緑のボタンを押すと、会計が行われる。

決済完了画面

図 4.2 のページで緑のボタンを押すと、図 4.3 画面に移行する。移行後は図 4.3 のとおり顧客の所持金から購入金額が差し引かれ、残高が表示される。また、所持金が入金金額が下回っていた場合は、所持金額が足りないと警告され決済は行われない。

カート1の中身
合計金額3316 ¥

[決済](#) [戻る](#)

商品名	価格	商品画像	個数
スッキリわかるSQL入門 [中山清喬]	3080 ¥		1個
森永製菓 ダース	118 ¥		2個

図 4.2. Web ページ カート内商品一覧

カート番号1
お支払金額6404 ¥
残高93596 ¥

[戻る](#)

図 4.3. Web ページ 決済完了

4.2 検証結果

V字モデルに従って、実装部分に関するテストを行った。次に、それぞれに対してテストを行うことで、要件を満たしているかどうかを確認した。以下にそれぞれのテストの項目結果を表で示す。

4.2.1. 単体テスト

3章では、単体テスト項目を作成して、詳細設計を満たしているかどうかを確認した。3章のテスト項目に対して、確認結果と、確認日というカラムを加えた。テスト項目に対して検証が成功すれば○を付ける。検証が失敗した場合は×を付ける。確認日は、テスト項目を検証した日時を記入した。

サーバ通信

サーバ通信の検証で行った単体テスト項目の検証結果を以下の表 4.6 に示す。

表 4.6. サーバ通信単体テストの項目

テスト	確認項目	確認結果	確認日
1	RaspberryPiからTCP通信で送られてきた複数の画像を正常に受信	○	12月24日
2	RaspberryPiからTCPで送られてきたフラグ(追加or削除)を正常に受信	○	12月24日
3	データ受信後、受け取った画像データをYoloに渡し結果を取得	○	12月24日
4	Yoloから受け取った座標値をもとにバーコード領域のみを切り取る	○	12月24日
5	pyzbarにバーコード画像を渡して番号を取得	○	12月24日
6	バーコード番号をカートDBに追加	○	12月24日
7	RaspberryPiに解析成功の成否をTCP通信で返す	○	12月24日

Yolo によるバーコード領域検知

Yolo におけるバーコード領域検知の検証で行った、単体テスト項目の検証結果を以下の表 4.7 に示す。

表 4.7. Yolo によるバーコード領域検知単体テストの項目

テスト	確認事項	確認結果	確認日
1	YoloをWindows10環境のGPUで動作できるようにコンパイル	○	12月24日
2	任意の学習データを学習させバーコード専用の識別機を作る	○	12月24日
3	Yoloをpythonプログラムで実行できるように修正	○	12月24日
4	OpenCV形式の画像をyoloに投げたらオブジェクト名と座標が がついになったリストが返ってくる	○	12月24日
5	カメラから15cmの距離でもYoloが検知率が80%以上を達成	○	12月24日

DB を使用した商品情報の管理

DBを使用した、商品情報の管理の単体テスト項目の検証結果を以下の表 4.8 に示す。

表 4.8. DB テーブル単体テストの項目

テスト	確認項目	確認結果	確認日
1	商品名、値段、バーコード番号、サムネイルURLの登録ができる	○	12月24日
2	商品名、値段、バーコード番号、サムネイルURLの削除ができる	○	12月24日

決済システム

決済システムにおける単体テスト項目の検証結果を、以下の表 4.9 に示す。

表 4.9. 決済システム単体テストの項目

テスト	確認項目	確認結果	確認日
1	カートごとに画面を分かれて表示させる	○	12月24日
2	カートの中にある商品の情報をWebページにリスト表示させる	○	12月24日
3	カート内にある商品で同じものがあれば個数を表示させる	○	12月24日
4	決済するとカート内にある商品がカート内DBから消える	○	12月24日
5	決済すると購入者である顧客の所持金が正常に変わる	○	12月24日
6	決済完了画面に残高、購入額を表示させる	○	12月24日
7	決済時に顧客の所持金が足りていなければ決算されないようにする	○	12月24日
8	商品が登録されているかの結果が返ってくる	○	12月24日
9	フラグから商品の追加、削除を判断してDBに反映する	○	12月24日
10	DBにある画像URLからWeb上に画像を表示させる	○	12月24日

4.2.2. 結合テスト

これらの単体テストの項目を満たしていることを確認することで、スマートモビリティレジシステムが、表 4.10 の結合テストの項目を満たしていることを確認する。結合テストでは、スマートモビリティレジシステムの中で筆者が実装を担当した、サーバ部分の項目が記されている。

表 4.10. 結合テストの項目

		確認内容	確認結果	確認日	確認者
サーバ側	受信	ラズパイからTCPで送られていた画像を正常に受信	○	12月26日	段原
		ラズパイからTCPで送られていた画像とフラグ(追加or削除)を正常に受信	○	12月26日	段原
		データ受信プログラムで受け取った画像データをYoloに渡し結果を取得	○	12月26日	段原
		Yoloから受け取った座標値をもとにバーコードのみを切り取る	○	12月26日	段原
		pyzbarにバーコード画像を渡して番号を取得	○	12月26日	段原
		バーコード番号をカートDBに追加	○	12月26日	段原
		エッジ側へ、サーバが画像受信から番号取得までが成功したかの結果を返す	○	12月26日	段原
		商品を置いたとき、商品の重量を取得する	○	12月26日	段原
	Yolo	Windows10でYoloがPythonで実行できる	○	12月26日	段原
		任意の学習データを学習させバーコード専用の識別機を作る	○	12月26日	段原
		デフォルトのYoloのpythonプログラムをクラスに変更する	○	12月26日	段原
		OpenCV形式の画像をyoloに投げたらオブジェクト名と座標がついになったリストが返ってくる	○	12月26日	段原
	バーコード判別	バーコード画像を番号に変換できる	○	12月26日	段原
	DB	カートごとに画面を分かれて表示させる	○	12月26日	段原
		カートの中にある商品の情報をリスト表示させる	○	12月26日	段原
		カート内にある商品で同じものがあれば個数を表示させる	○	12月26日	段原
		決算するとカート内にある商品がカート内DBから消える	○	12月26日	段原
		決算すると購入者である顧客の所持金が正常に変わる	○	12月26日	段原
		決算完了画面に残高、購入額を表示させる	○	12月26日	段原
		決算時に顧客の所持金が足りていなければ決算されないようにする	○	12月26日	段原
		商品が登録されているかの結果が返ってくる	○	12月26日	段原
		フラグから商品の追加、削除を判断してDBに反映する	○	12月26日	段原
		DBにある画像URLからWeb上に画像を表示させる	○	12月26日	段原

4.2.3. 総合テスト

結合テストの結果を確認し、各モジュール間でのデータの受け渡しが、正常に行われていることを確認した。最後に、表 4.11 に総合テストの項目を示す。総合テストでは、ユーザが買い物を始めてから、決済するまでの流れをシナリオとしている。このシナリオでは、ユーザの残高は購入金額を下回らないと仮定する。また、決済前に買い物自体をキャンセルすることはないと仮定とした。

表 4.11. 総合テストの項目

シナリオ	確認の流れ	確認結果	確認日
買い物	1-1 商品をカゴに入れ、超音波センサ反応時画像を撮りLEDを点灯させる。	○	1月7日
	1-2 重量センサより商品追加・削除の情報をサーバへ送る。	○	1月7日
	1-3 バイナリーデータを画像とフラグに変換する。	○	1月7日
	1-4 Yoloを使用して画像の中のバーコードのみを切り取る。	○	1月7日
	1-5 バーコード番号を解析する。	○	1月7日
	1-6 商品のバーコード情報を確認する。	○	1月7日
	1-7 商品を商品DBに追加・削除する。	○	1月7日
	1-8 商品を正常に商品DBに追加できたかどうか結果をLED点灯にて通知する。	△	1月7日
決済	2-1 カートを選択する。	○	1月7日
	2-2 商品を確認し、決済を行う。	○	1月7日

第 5 章

システムの評価・考察

本章では、V字開発モデルに従って実装したシステムを評価し、本システムに対する利点と問題点について述べる。このシステムの利点は、Yolo を使用することで高い精度でバーコード領域特定を行えることである。複数の商品を、同時に読み取る場合であっても、Yolo を使用すれば容易にバーコードの検知を行うことができる。バーコード識別可能なモデルの学習も画像 2000 枚程度で、高い精度を出すことができる。

表 5.1. 実行環境

Web カメラと対象の距離	精度
10cm	98～99%
15cm	92～93%
20cm	95～96%

上の表 5.1 より、対象から距離が離れても極めて高い精度を保っていることがわかる。Yolo の学習にかかった時間もおよそ 10 時間ほどであり、比較的短時間で学習が終了した。高い精度を出すことができる理由として、学習する際に各バーコード画像がもつ特徴の差が少ないということと、判別する対象が 1 つだけであることが挙げられる。この特徴の差が少ないというのは、機械学習において精度向上と学習時間短縮に繋がる [8]。例えば、猫の識別の場合、猫にも様々な種類があり、精度を高めるには様々な種類の猫のサンプルを集める必要がある。しかし、フォーマットの定められて

いるバーコードなら、多くのサンプルを集める必要がない。また、判別する対象が 1 つだけであることは、学習時間短縮につながる。判別する種類が多くなると一般的 [8] に、機械学習で物体を判別する際により複雑なネットワークと学習用の画像データを多く必要とする。また、高性能な GPU が必要になる。

一般的な大規模店舗では、数万種類の商品を扱っていることもある。もし、Yolo をバーコード識別に使用するのではなく、商品個別の判別に用いる場合、数万種類を高い精度で判別できるネットワークを構築する必要がある。さらに、その学習を行うことのできる性能の機材も必要となる。もし、そのようなネットワークを実現できたとしても、新しい商品が追加されるとその商品の判別を行えるように画像データを収集し、再学習する必要がある。今の技術及びコストを考慮すると、これらを行うのは非現実的である。そこで、どの商品でもほとんどフォーマットが変わらないバーコードの識別を行うことにした。

今回実装と検証を行って、一番の問題点となったのが Web カメラが撮影した商品の距離によって検知率が大きく異なる点である。今回使用した Web カメラはロジクール製の C615 モデルである。このモデルを選ぶ前に同メーカーの C270n モデルを使用していたが、商品から 5cm 程の近距離でバーコードを撮影しても、バーコード番号の識別が不可能であった。しかしながら、Yolo による領域検知は行うことができた。C270n モデルの画質では番号判別ができなかったため、C615 モデルに変更した。テストを行ったところ、約 10cm の範囲の近距離ならば、pyzbar で正確に識別できることが、判明した。さらにテスト行い距離を 10cm 以上伸ばしたところ、画像のバーコードがぼやけてバーの境界があやふやになっている状態だったため、検知しないことが分かった。バーの境界を明確にするために画像に対して 2 値化処理とノイズ除去を行ったが、結果は変わらなかった。代わりに、iPhone5c を使用して 15cm の距離で撮影した画像でテストしたところ、pyzbar での識別を行うことができた。本システムは Web カメラの画質によって識別精度が大きく影響されると結論付けられる。

第 6 章

まとめ

本論文では、商品識別システムの設計および実装を部分的に行った。システム設計に関しては V 字モデルに従い、UML を使用して定義づけた。実装は試作の観点から、システムを稼働させるにおいて最も重要となる点のみを実装した。開発はグループにより分担しておこなった。グループで実装を円滑に行うために GitHub を使用して行った。実装したシステムの評価に関しては単体テスト、結合テスト、総合テストに基づいてそれらの項目を満たすかどうかを確認した。テストの行程で発生した問題は、グループで共有して担当部分以外でもお互いにサポートして問題解決をおこなった。今回開発したシステムは、高い安定度を維持することができれば、既存のセルフレジと比べて極めて安価なリソースでサービスを提供することが可能であることを確認した。

謝辞

本研究を進めるにあたり、懇篤な御指導、御鞭撻を賜りました本学高橋寛教授に深く御礼申し上げます。

本論文の作成に関し、詳細なるご検討、貴重な御教示を頂きました本学樋上喜信教授ならびに王森レイ講師に深く御礼申し上げます。

また、審査頂いた本学岡野大准教授ならびに宇戸寿幸准教授に深く御礼申し上げます。

最後に、多大な御協力と貴重な御助言を頂いた本学工学部情報工学科情報システム工学講座高橋研究室の諸氏に厚く御礼申し上げます。

参考文献

- [1] 平成 28 年版 情報通信白書 | 人口減少社会の到来, 総務省,
<https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h28/html/nc111110.html>,
2016-7
- [2] セミセルフレジのメーカーまとめ。価格・製品特徴比較【2019 年版】
<https://rejichoice.jp/semi-self-regi/> 2019-3-11
- [3] 阪田史郎, 高田広章, 組込みシステム, 株式会社 オーム社, 2006
- [4] 株式会社 オージス総研, かんたん UML[増補改訂版], 株式会社 翔泳社, 2003
- [5] YOLO:Real-Time Object Detection <https://pjreddie.com/darknet/yolo/>
- [6] pyzbar <https://github.com/NaturalHistoryMuseum/pyzbar> 2019-2-21
- [7] 詳解 OpenCV 3 — コンピュータビジョンライブラリを使った画像処理・認識 2018-
5-26
- [8] ゼロから作る DeepLearning オライリージャパン 2018-9-28