

# 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                | 2  |
| 2.1   | 諸定義 . . . . .     | 2  |
| 2.2   | システムの概要 . . . . . | 4  |
| 第 3 章 | システムの設計           | 5  |
| 3.1   | 要求定義 . . . . .    | 5  |
| 3.2   | 設計 . . . . .      | 5  |
| 第 4 章 | 実装内容・検証結果         | 10 |
| 4.1   | 実装・実験環境 . . . . . | 10 |
| 4.2   | 検証 . . . . .      | 16 |
| 第 5 章 | システムの評価・考察        | 20 |
| 第 6 章 | まとめ               | 22 |
|       | 謝辞                | 23 |
|       | 参考文献              | 24 |

## 第 1 章

### まえがき

本論文では、番号からラズベリーパイと各種センサとサーバを組み合わせて商品識別システムを構築した。皆さんもご存じの通り現在の日本は少子高齢化の流れで人的資源が減少している。総務省の調べでは総人口が 2008 年をピークに減少している [1]. 人的資源の減少という問題は私たちの身近なスーパーマーケットや小売店にも表れており、その対処法として決済の一部をユーザに一部負担してもらうセルフレジなどの導入が進んでいる。また Amazon などの大企業では無人店舗 AmazonGo[2]. をアメリカの一部地域で展開しているがそのサービスを稼働させるには高い技術力と資金力が必要となる。先ほど述べたセルフレジの金額も登録機と精算機を合わせて平均で 300 万を超えるものが多い [3]. 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が負担も大きい問題がある。そこでこのコスト面で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い安価なシステムの構築を本研究の目的とした。

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 図を用いて定義しそれをもとに開発を行う。実装に関してはエッジ処理側とサーバ処理側で担当を分けた。

本論文は以下のような構成をとる。第 2 章では用語、実験環境、システム概要の説明を述べる。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と今後の課題を示す。

## 第 2 章

## 準備

本章では本文中に使用する用語、システム概要について述べる

### 2.1 諸定義

#### V 字モデル

V 字モデルとはシステム開発を要求分析、基本設計、詳細設計、実装に分け上から順にそれぞれに対して検証、テストを行いながら上から順に時間軸に従って行っていく。細かく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。

#### UML(Unified Modeling Language)

UML とは各開発工程で利用すべき図面の標準化されたモデルである。UML で一番重視されていることは、単純で分かりやすいという軸と十分実用に使えるだけの強力な表現方法を持つ軸を最適に組み合わせて言語設計をすることである。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。[5]

#### ユースケース図

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。

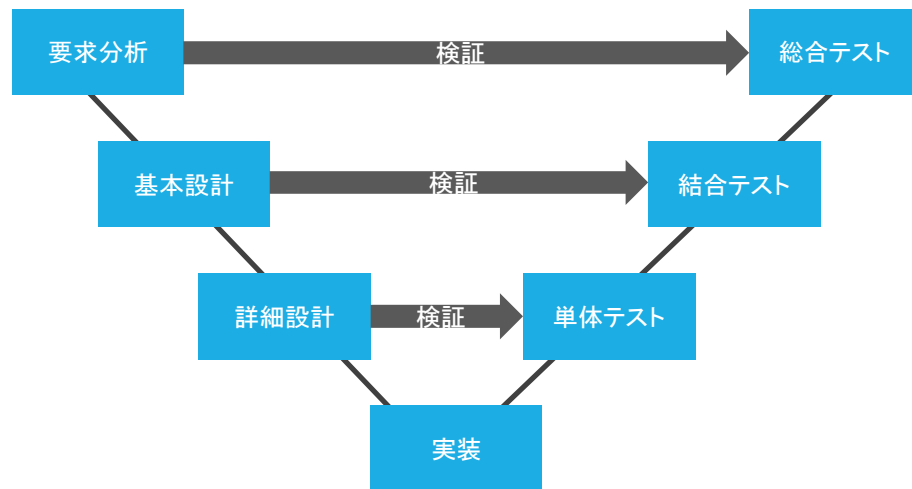


図 2.1. V字モデル

## シーケンス図

シーケンス図は、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。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していく。

## クラス図

クラス図はモデルの静的な構造を示す図。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していく。

## 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なネットワークを目的としている。Web カメラでのリアルタイム検出を行うこともできる FPS を誇る。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。今回は画像からバーコードの位置を特定するために使用した。[6].

## pyzbar

バーコード画像データを解析して数字を識別するライブラリである。[7].

## 2.2 システムの概要

システムでは、買い物かごにエッジ処理を担当する RaspberryPi と各種センサを設置する。エッジ側では商品の特定に必要なデータをセンサ類を用いて取得する。データを取得したらサーバ側に送信し、そこで画像データを処理して商品の特定を行う。システムの流れを以下の図 2.2 に示す。

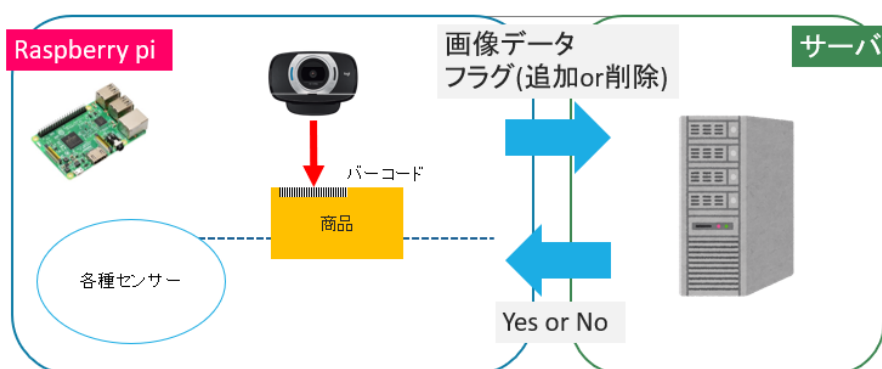


図 2.2. システムの流れ

左側の RaspberryPi 側で商品に関するデータをサーバに送信する。画像のフラグは送信された画像の商品がかごから追加されるものなのか削除されるものなのかを判断するのに使用する。サーバはデータを受け取ったのちバイナリーデータから画像データに戻す。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 に識別が成功したフラグを返信する。

## 第 3 章

# システムの設計

本章では V 字モデルによる要求定義および設計について述べる。本章の構築は以下のとおりである。3.1 節ではユースケース図を用いて要求定義を説明する。3.2 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。

### 3.1 要求定義

ユーザがシステムに求める機能や動作を決定する。以下に要求定義をまとめた図 3.1 を載せる。赤枠で囲っている部分が担当部分になっている。3.1 では、ユーザがシステムに対して動作を行った際にどのような振る舞いをするか記載してある。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。解析システムでは画像から商品の特定やカゴ DB への操作を行う。この解析システムはサーバで動作する。カゴ DB ではカート内にある商品の管理を行う。ここで管理されている情報が最終的な決済システムで利用される。ユーザはカートを返却すると決済システムが動作し自動で決済システムが動作し顧客の所持金からカート内の商品合計金額がひかれる。

### 3.2 設計

以下に基本設計、詳細設計を示したシステムのクラス図 [4] を図 3.2 に載せる。

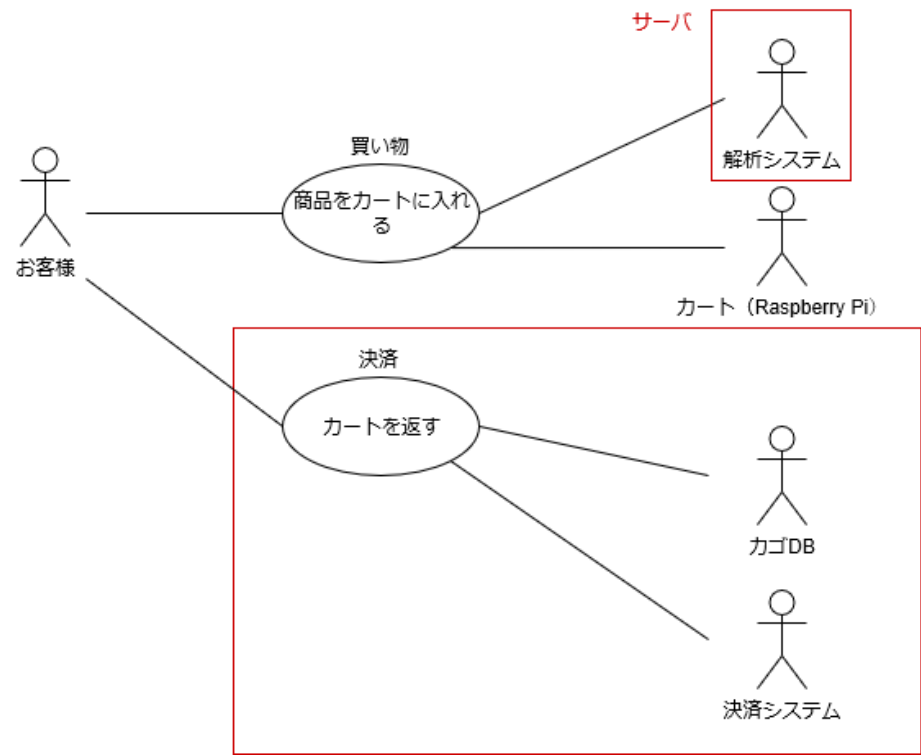


図 3.1. ユースケース図

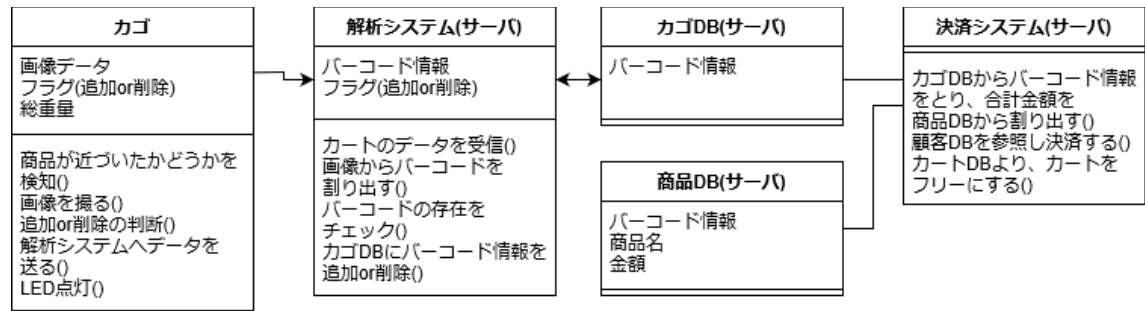


図 3.2. クラス図



図 3.2 は大きく分けてカゴ、解析システム、カゴ DB、商品 DB、決算システムをクラスとしたときにそれぞれが持つ属性と操作を表している。カゴはエッジ側は処理を担当し、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したりユーザに解析が成功したかの通知をするためのクラスをまとめたものである。解析システムはカゴクラスから送られてきたデータの受信、解析とカゴ DB への操作をクラスにしたものである。カゴ DB は現在ユーザが購入予定の商品に何があるのかを管理する。商品 DB には商品のバーコード番号と商品名、値などの商品自体の情報が管理されている。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。最後に決済システムは最終的にユーザが購入する際に必要な機能をクラスにしている。

3.2 に記述されているシステムの機能について説明していく。

- カゴ カゴの属性である画像データは商品が近づいたかどうか検知する操作によって取得する。画像データ属性とフラグ (追加・削除) の 2 つのデータの取得が完了すると解析システムへデータを送る。ここでいうフラグの追加の役割はユーザがカゴに購入予定の商品を入れる場合にしようする。削除はすでにカゴ内にある商品の購入をやめる場合に使用する。このフラグ (追加・削除) の判断には重量センサを用いて行う。重量センサから取得できる総重量が増えた場合は追加すると判断し、減少した場合は削除すると判断する。LED 点灯は解析システム側で解析が成功した場合にユーザへの通知としての役割がある。
- 解析システム カゴクラスが解析に必要な画像データとフラグ (追加・削除) を送信すると解析システムがデータを受信して解析が始まる。画像データからバーコード番号を割り出すことに成功すると一緒に受信したフラグを使用してカゴ DB にバーコード番号を追加する。ただし存在しない番号を追加してしまわないように DB に追加する前に商品 DB に該当するバーコード番号があった場合のみ追加を行う。また解析に成功しても失敗してもカゴへ成否の結果を送信する。
- カゴ DB カゴ DB はユーザの購入予定の商品のバーコード番号の管理を行う。この DB は解析システムによって操作される。また決済システムによって決済が終了した場合はデータは削除される。
- 商品 DB この商品 DB はシステムの動作中には追加・削除などの操作は行われ

ず、参照のみ行われる。この DB は店側が新商品の追加や値段の更新を行いたい場合に操作される。

- 決済システム 最後の決済システムが行うのはユーザが決済を行うときに機能する。ユーザが商品を出し入れする時点でカゴ DB に購入予定の商品の情報が格納されているため決済時にカゴの中にある商品を 1 つずつスキャンする必要はない。つまり決済が一瞬で完了することになる。

次にシーケンス図を用いてシステムの流れを説明する。以下の図 3.3 にそって説明する。赤枠でカッコている部分が実装担当である。図??はエッジ処理側、サーバ側、Web ページ側の大きめに 3 つに分かれている。左から右の順に処理が流れていく。上部の四角の項目はオブジェクトと呼ばれる。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。シーケンス図から読み取れるようにシステムはユーザからカートへ商品が渡され、エッジ側であるカートで処理が行われる。エッジ処理が終わるとサーバへ情報が伝達され解析が始まる。解析結果を DB とエッジに送信する。結果は Web ページへ反映される。エッジ側では DB の操作は行われない。

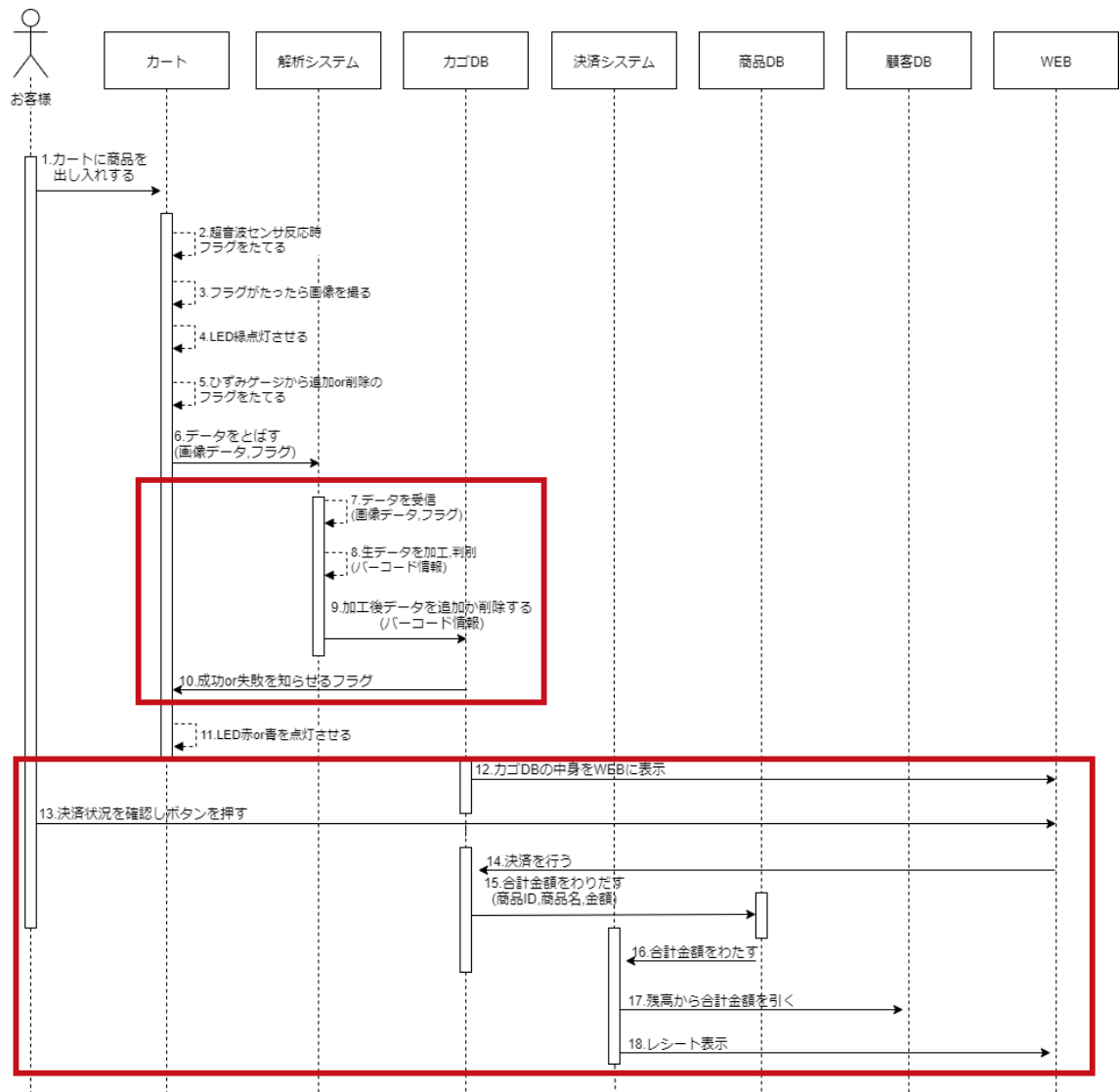


図 3.3. シーケンス図

## 第 4 章

# 実装内容・検証結果

本章では V 字モデル開発に従って実装を行った。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

実装に至っては 2 人のグループで行いエッジ処理側は真鍋樹が行った。ここではサーバ側の実装について述べる。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、サーバどちらも Python3 を使用した。

#### 実験環境

実験環境で使用したものを以下の表 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 | RaspberryPi3B(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。クライアントはセンサが反応したらカメラを起動し複数の画像を撮影する。つまりサーバに送られる画像データは1回につき1枚ではない。つまりクライアントが画像データを送信するデータサイズが不明のため、サーバ側では送信されたデータのどこからどの部分が1つの商品画像データになるのか判断できない。そこで、クライアントは画像データを送信する前に画像データの合計サイズを送信する。サーバは合計サイズ分だけデータを受信すれば続けて2回目のデータ送信が来ても1回目のデータと区別することができる。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[8]. のフォーマットに変換しなおしている。

## Yolo によるバーコード領域特定

Yolo を使用した理由は後述する pyzbar[7]. の識別精度にある問題があったためである。pyzbar[7]. は近距離で撮影したバーコードの画像の認識はうまくいくが、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ってしまう認識がうまくいかないためと思われる。そこで Yolo を使用し、画像からバーコードの部分のみの座標を取得する。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期待されたが、残念ながらラズパイの性能では 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かっ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

カゴの中の商品を管理と、商品自体の情報の管理に DB を使用した。商品 MariaDB を使用した。カゴ DB の構造を以下に示す。初めに商品自体のデータを管理する item テーブルを 4.1 に示す。jan は JAN コードのことであり、商品識別番号のことである。title は商品名のことである。price 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できるようにしている。

|                          | # | 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 |

図 4.1. item テーブル

次にカートのテーブル cart テーブルの構造を図 4.2 に示す。id 行の固有番号を示すためにある。この番号をでカートごとの区別を行う。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。この番号があることで複数運用した際も区別することができる。


|                          | # | 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     |      |                |

図 4.2. cart テーブル

顧客テーブルの構造を図 4.3 に示す。この顧客テーブルはシステムの動作テストを行う際に必要だったため作成した。動作用に仮の顧客情報を管理する。id は顧客の固有番号を示し、balance は顧客の所持金額を示す。


|                          | # | 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     |      |                |

図 4.3. customer テーブル

## 決済システム

設計では決済システムが動作するのはユーザが退店する際に自動で行われることになっているが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 Web ページを作成して代用している。ページ作成には PHP を使用している。以下に Web ページでの動作の流れを示す。

### カート画面

本来はユーザがこのような画面を操作することなく自動で決済が行われるのでこれは動作テスト用に作成している。以下の図 4.4 は初めにどのカートを使用しているかを選択するためにある。

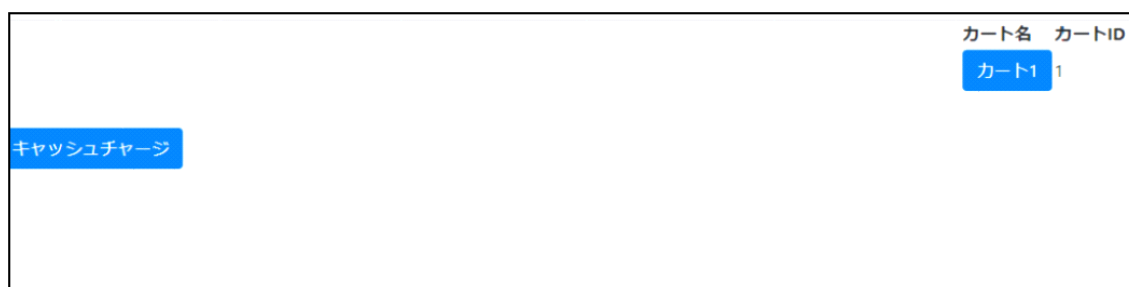


図 4.4. Web ページ カート画面

### カート内商品一覧

こちらの図 4.5 はカート内にある購入予定の商品の一覧を表している。表示内容は商品名と価格、商品画像と個数の 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 4.5 のページでボタンを押すと図 4.6 画面に移行し、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、所持金が購入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会計は行われない。



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 [ 中山清希 ] | 3080 ¥ |  | 1個 |
| 森永製菓 ダース              | 118 ¥  |  | 2個 |

図 4.5. 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 4.6. Web ページ 決済完了

## 4.2 検証

V字モデルに従って4.1節で述べた実装部分に関する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する。以下にそれぞれのテスト項目を表で示す。

### サーバ通信

サーバ通信の検証で行った単体テストを以下の表4.7に示す。

| 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             |

図 4.7. サーバ通信単体テスト

### Yolo によるバーコード領域特定

Yolo におけるバーコード検知の検証で行った単体テスト項目を以下の表 4.8 に示す。

| 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<br>がついになった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                      |

図 4.8. Yolo によるバーコード領域特定単体テスト

## 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を以下の図??に示す。

| 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 |

図 4.9. DB テーブル単体テスト

## 決済システム

決済システムにおける単体テストを以下の図 4.10 に示す

| 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      |

図 4.10. 決済システム単体テスト

これらの単体テスト項目を満たしていることを確認することにより、システムが図4.11の結合テストの項目を満たしていることを確認する。結合テストではシステムの中で筆者が実装担当したサーバ部分の項目が記されている。

|      |         | 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になっ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                       |

図 4.11. 結合テスト

結合テストの内容を確認し、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する。最後に図4.12に総合テストの内容を示す。総合テストではユーザが買い物をはじめ最後に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予定金額を下回らないと想定し、また買い物キャンセルすることはないという想定になっている。

| 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。                       |

図 4.12. 総合テスト

## 第 5 章

### システムの評価・考察

本章では V 字モデルに従って実装したシステムを評価することにより、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として Yolo を使用することで高い精度でバーコード領域特定ができることである。複数の商品を同時に読み取りたい場合であっても 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。バーコード識別ができるモデルの学習も画像 2000 枚程度での学習で高い精度が出る。

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 を見てもわかる通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。学習にかかった時間もおよそ 10 時間ほどである。これの高い精度を出すことができる理由として、判別する対象が 1 つだけである点と学習するバーコード画像自体の違いが少ないという点もある。この違いが少ないというのは例えば猫の識別の場合猫といって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が、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。もう 1 つの判別する対象が 1 つだけであることで得られる利点としては、機械学習で物体を判別する際に判別する種類が多くなると一般的により複雑なネットワークと学習用の画像データを必要とした、強力な GPU が必要になる。一般的な大型販売店などでは数万種類の商品を扱っている。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断をする場合、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りまたその学習を行える性能の機材も必要となる。またそのような判別機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。今の技術では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 を使用している。こちらのモデルを選ぶ前に同メーカーの C270n を使用していたが 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。ここでいう識別は pyzbar による番号の識別であり Yolo による領域判定は検知していた。C270n モデルの画質では判別が不可能と判明したため C615 モデルに変更したところ約 10cm の範囲の近距離ならば pyzbar で正確にすべて識別することがテストを行い判明した。しかし、C615 モデルで 10cm 以上離すと検知しなくなった。画像に対して 2 値化処理とノイズ除去を行ったが、元の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結果は変わらないという結果に至った。代わりに iPhone5c で 15cm の距離で撮影したバーコードを映した動画をテストしたところ pyzbar での識別をある程度行うことができたため、本システムは Web カメラの画質によって大きく識別精度が変わる問題点がある。

## 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。実装は時間的な兼ね合い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、開発はグループで行ったので分担した。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することにした。テストを実行する過程で発生した問題はグループ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動作が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り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准教授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] Amazon GO でミライ体験! レジ無し AI コンビニの仕組み、技術と課題  
<https://orange-operation.jp/posrejihikaku/self-checkout/10331.html> 2017-6-13
- [3] 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
- [4] 阪田史郎, 高田広章, 組込みシステム, 株式会社 オーム社, 2006
- [5] 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
- [6] YOLO:Real-Time Object Detection <https://pjreddie.com/darknet/yolo/>
- [7] pyzbar <https://github.com/NaturalHistoryMuseum/pyzbar> 2019-2-21
- [8] 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