サーバ

サーバ通信：

1.クライアントから送信されるデータサイズとタイミングが不明のため、初めに固定長で画像データの合計サイズを固定長で送信する。クライアント側は受け取ったデータサイズをもとに画像データを受信する。

2.サーバの通信はスレッドを使用して複数のクライアントが接続しても稼働するようにしている。

3.クライアントから送られてきた画像データはバイナリー形式になっているのでOpencvで操作できるように変換している

4.バーコードが識別で来たかどうかの結果をクライアント（ラズパイ）を通してユーザに通知する必要があるため、バーコードの読み込みの成否を返信する。

Yolo:

1. Yoloを使用した理由は後述するpyzbarの識別精度にある問題があったためである。

pyzbar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い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。

1. Yoloの実行はサーバ側で行う。当初クライアント側であるラズパイでYoloを実行すればサーバは不必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。
2. Yoloを使用するほかの理由として商品のカテゴリわけができる点である。カテゴリわけができることでバーコード識別際に誤った結果を返すことを軽減できる。どういうことかというとバーコードを認識する前に大まかに商品のカテゴリが分かれば正解であるバーコード番号を絞ることができるためである。例えばカテゴリ識別は正しく認識し、バーコード認識が間違った場合があるとする。カテゴリ識別では飲料になっているがバーコード番号識別で割り出した商品がブロッコリーだった場合、誤認識であると検知することができる。

バーコード認識:

1.

2.

3.

DB:

1.ユーザごとのカート内の商品を管理する必要があるため、DBを使用する。カート内の商品の管理以外にも商品自体の値段やタイトル、画像そのものも管理する。

決済:

1.最終的にユーザはレジを通らず決済するため