平成 27 年度　特別研究報告書

冷蔵庫内食品の自動認識及び管理の考察

龍谷大学　理工学部　情報メディア学科

T120430　川村 紘菜

指導教員　三好 力 教授

内容梗概

普段買い物に行く際冷蔵庫の中身を覚えておらず，つい買いすぎてしまったり，買い忘れてしまったりすることがある．冷蔵庫管理に関するアプリケーションや関連研究はいくつか存在するが，どれも手動で登録するものや，音声対話を活用したものなど，自動管理ではない．本研究は，既存技術より便利な冷蔵庫内の食品の自動管理を実現させるためのものである．完璧な自動化は現段階では困難であるため，自動化が困難な部分は既存技術の音声対話等をうまく利用することによってよりスマートなシステムを構築できると考えた．提案手法として，冷蔵庫内に小型のカメラを取り付け，撮影した動画像から静止画抽出，背景差分及びテンプレートマッチングを行い，食品が認識されるかどうかを研究することで，自動管理としての機能を果たせるかどうかを考察した．

目次

第１章　緒論

普段スーパーマーケット等に買い出しに行く際，冷蔵庫の中に入っている食材を忘れてしまい，つい同じ食材を購入してしまったり，あると思い込んで購入せずに帰宅し，実際冷蔵庫内に食材はなく，調理に困ったりするという問題がある．予め，買うものをメモすることで買い忘れ等は阻止できるが，わざわざ冷蔵庫の中身を確認し，メモを取っていくことは手間である．加えて中身を確認するためだけに長時間冷蔵庫のドアを開けっ放しにすることはエコではない．これらの問題を改善するために冷蔵庫内の食材を自動で管理し，普段持ち歩いているスマートフォンや携帯等の端末に管理状況を通知する方法が挙げられる．こうすることで，いつでもどこでもユーザが冷蔵庫内の食材を把握することができ，より効率の良い買い出しが出来るようになるのではないかと考える．既にいくつか冷蔵庫管理システムが存在しているが，どれも完璧な自動化は成されていない．本研究ではこれを実現するための前段階として，冷蔵庫内に小型のカメラを取り付け，撮影した動画像から出し入れした食品が自動で認識できるかどうかについて，その手法を考察した．

第 2 章　関連研究及び既存技術

2.1　画像処理と音声対話による食品管理システムについて

冷蔵庫の中身に対してユーザからの質問に答えるサービスとして，食品全般を対象とした画像処理と音声対話による食品の認識を行うという研究がある[1].

システムの動作は大きく 3 つの場面に分かれており，ユーザの発話によってどの場面かを判断している．以下に説明する．

(1)食品を入れる場面

冷蔵庫の画像から食品の位置や形，状態(隠れているなど) を自動的に認識する．また，音声によって名前や場所をユーザから入力して貰う．その後，食品のデータ(位置，形，名前など) を登録する．

(2)食品を取り出す場面

取りだされた後の冷蔵庫の画像から，取り出された食品を認識する．画像処理による特定が難しい場合には，ユーザとの対話によって取られた食品を特定する．その後，食品のデータの一覧から取られた食品の削除，及び食品を取った際に周りの食品が移動した場合には，食品のデータの位置の変更を行う．

(3)ユーザが食品の確認や食品を取る依頼を行う場面

蓄積した食品のデータを基にユーザとの対話を行い，食品の場所や存在するかどうかといった質問に答える．

冷蔵庫内の食品の認識法は，1 つの食品の出し入れが行われるたびに冷蔵庫の画像を保存する．まず，1 つ前の時系列の画像と現在の画像との差分を取り，差分が大きい範囲を食品の認識の処理を行う範囲(以降処理範囲とする) とする．次に，処理範囲内で，冷蔵庫全体の照明の変化に対する明るさの補正，カメラのずれに対する対処を行う．その後，対処した画像を用いて改めて背景差分を行う．最後に影の除去を行い，食品の領域を検出する．具体的には，最初に食品を入れた際に食品のモデルを登録する．その後は，食品のモデルを用いて入れた食品の認識を行い，順次食品のモデルを登録していく．既に入っていた食品が移動した場所を処理範囲内でテンプレートマッチングを行い特定する．似た色を持つ食品が入れられた場合や，記録している代表の色以外の色を持つ模様が回転し現れた場合には，移動先として可能性のある複数の位置を移動した候補の位置として，候補全てでその位置の正しさを評価する．評価によっては，ユーザとの対話を行い，移動した場所を選択して貰うことで食品が移動した場所を正確に把握する．

2.2　荷重特徴を用いた食材同定について

食材管理の方法として，食材が棚のどこに置かれたか，つまり食材の位置情報を食材同定に利用するという研究がある[2].

食材の位置は，棚に複数の秤を設置し，その秤の荷重バランスによって表される．具体的には四隅に秤(荷重センサ)が付いた板(荷重センサボード)を冷蔵庫の棚に設置する．その荷重センサボードに食材を置いたときの荷重センサの値の変化を見ることで，ボード上のどの位置で荷重に変化が生じたか，すなわち食材が置かれた，もしくは取られた位置に関する情報を取得する.

冷蔵庫に食材が置かれる(もしくは取られる)前後での 4 つの荷重センサの変位をΔz＝(Δz_1，Δz_2，Δz_3，Δz_4)とする．Δzを重さと位置を表す特徴“荷重特徴”として同定に用いる．食材が置かれたときと取られたときとでΔzの向きが正反対であることを考慮すると，ある食材が置かれたときの荷重特徴Δz^{IN}とある食材が取られたときの荷重特徴Δz^{OUT}との非類似度はΔz^{IN}と$-\Delta z^{OUT}$の 2 つのベクトル間の距離により与えられる．よって，Δz^{OUT}との非類似度が最小となるΔz^{IN}を探索し，これをΔz^{OUT}に対応する食材とする.

サイズ 550mm×290mm，厚み 6mm，重さ 2.5kg，耐荷重 5kg の強化ガラスの下部の四隅に荷重センサを設置し，A/D コンバータを用いて，荷重センサからの電圧出力を数値データに変換をし，そのデータを処理する.

2.3　既存の冷蔵庫管理アプリについて

2.1 と 2.2 で述べた関連研究の他に，既に冷蔵庫内の食材を管理するシステムがアプリケーションとして利用されている[3]．これらの食品管理アプリは冷蔵庫の中に入れた食材の種類や，賞味期限の管理，冷蔵庫にある食材で作れるレシピを確認することができる．

アプリケーションの利用方法は主に 3 つある．以下に記す．

1. 冷蔵庫に入れた食材を手入力で登録する
2. アプリ内にある検索画面から食品を探し登録する
3. 食品についているバーコードを読み取り登録する

なお，賞味期限や冷蔵庫に入れた日時等は全て手入力となる．また，食材を冷蔵庫から取り出した際に関しては上記で述べた 1~3 の方法で登録する．冷蔵庫管理アプリの例を図 2.1 及び図 2.2 に示す．

図 2.1：冷蔵庫管理アプリ(リスト)

図 2.2：冷蔵庫管理アプリ(バーコード読み取り)

2.4　既存の音声対話による冷蔵庫について

既存技術として他に，2015 年 10 月 6 日にシャープが発表したプラズマクラスター冷蔵庫「SJ-TF50B」がある[4].

この冷蔵庫は対話をしながら庫内の食品管理ができ，食品を冷蔵庫に入れる際に声で食品名を登録しておくと，冷蔵庫が使用期限が近づいた食品を教えることが可能である.「期限が近い食品でなにが作れる?」と尋ねると，レシピを提案する機能も持つ.

専用のスマートフォン向けアプリ「ココロボ～ド」を利用すれば，冷蔵庫内の食品量や使用期限，保存場所といった詳しい情報も登録可能である．保存場所を登録することで「マスタードどこにいったっけ?」と聞くと「小物ポケット右にあるよ」などと回答する．冷蔵庫が音声で認識できる食品数は約 1,650 個で，提案メニュー数は約 220 種類である.

第 3 章　既存技術に対する問題点

第 2 章で述べた関連研究及び既存技術にはいくつか問題点が考えられる．

まず 2.1 の画像処理と音声対話による食品管理システムについて，一つ一つの食品がとり出された際に自動で撮影を行うと記述してあるが，冷蔵庫を開けた状態で何回も食品を出し入れした際にどのようにして一つの食品を入れたと認識させる方法が書かれていないため，カメラの自動撮影は極めて困難であると考える．また，音声対話に関しては冷蔵庫に向かって言葉を発することは，抵抗もあり，面倒であると考える．他にも，最初に入れた食品をモデルにしていくだけでは向きや形が変化した際に対応できないのではないかという問題点が挙げられる．

次に 2.2 の荷重特徴を用いた食材同定については，位置情報と重さ特徴だけでは正確な食材の同定が非常に不安定であると予測される．また、食材を取り出し，半分利用してまた冷蔵庫内に戻した場合，重さに変化が生じ，また位置も元の場所に戻すとは限らないため，食材の同定はほとんど不可能ではないかと考える．

次に 2.3 の既存の食品管理アプリについては，機能に関しては便利であるが，一つずつ食品を手入力していくため，非常に面倒であり，管理が続かない場合がある．また，野菜等のバーコードがない食品も多数存在するためやはり手入力の操作が必要であり，不便であると考える．

最後に 2.4 の既存の音声対話による冷蔵庫については，2.1 と同様音声認識であるため，やはり自ら冷蔵庫に向かって声を発することになるため面倒であると考える．また，2.3 のアプリ同様，食品すべてを音声で入力するため，面倒であり，管理が続かない恐れがある．

以上の問題点から，より便利な管理を目指すためには普段冷蔵庫に物を出し入れする動作だけで自動で中身を認識できることが望ましい方法であると考える．

第 4 章　提案手法

第 3 章で述べた問題点を解決する策として，2.1 にあげた関連研究の改良法を検討した．冷蔵庫内をカメラで撮影するのではなく，冷蔵庫の扉が開いている間は動画を撮り，撮れた動画像を解析するという方法を挙げる．

4.1　web カメラによる冷蔵庫内の撮影

カメラは web カメラ等の小型カメラを使用し，冷蔵庫の見やすい位置に設置する．正面方向からの位置と上方向からの位置と 2 台以上設置する方が好ましい．動画の撮影は，冷蔵庫を開けた時につく庫内灯センサを利用する．庫内灯がついている間，つまり冷蔵庫が開いている間のみ録画を行う．庫内灯が消える，つまり冷蔵庫が閉まった時は撮影を中止するよう設定する．

4.2　動画像内の静止画抽出

4.1 で撮影した動画を解析する．つまり，一つの食品が出し入れされる前後のフレームを抽出する．具体的には，撮影した動画像から動きが発生する前と動きがなくなった後の静止画を自動で抽出する．静止画像を抽出する手段として，OpenCV を使用した．OpenCV とはインテルが開発・公開した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．最新バージョンは現段階で 3.0.0 であるが，本研究ではサンプルの多い 2.4.9 を使用した．

まず，撮影した動画像からある程度のフレーム間隔で静止画を抽出する．抽出された順に静止画に番号をつけ，前から順番に 2 枚ずつ静止画を比較する．比較内容は，1 ピクセル毎にどれくらい変化がおきたか，差分を取り，値を絶対値にすることで 1 ピクセル毎の変化量を調べる．各ピクセルの差分値が設定した閾値以上となった場合，ピクセルをカウントしていく．全てのピクセルを比較し，閾値以上となったピクセル数が画像全体の 2%となったときの静止画像を保存していく．これで冷蔵庫内に食品を出し入れする直前と直後のフレームを静止画像として自動で保存することが可能であると考えた．

図 4.1 に静止画抽出のフローチャートを示す．

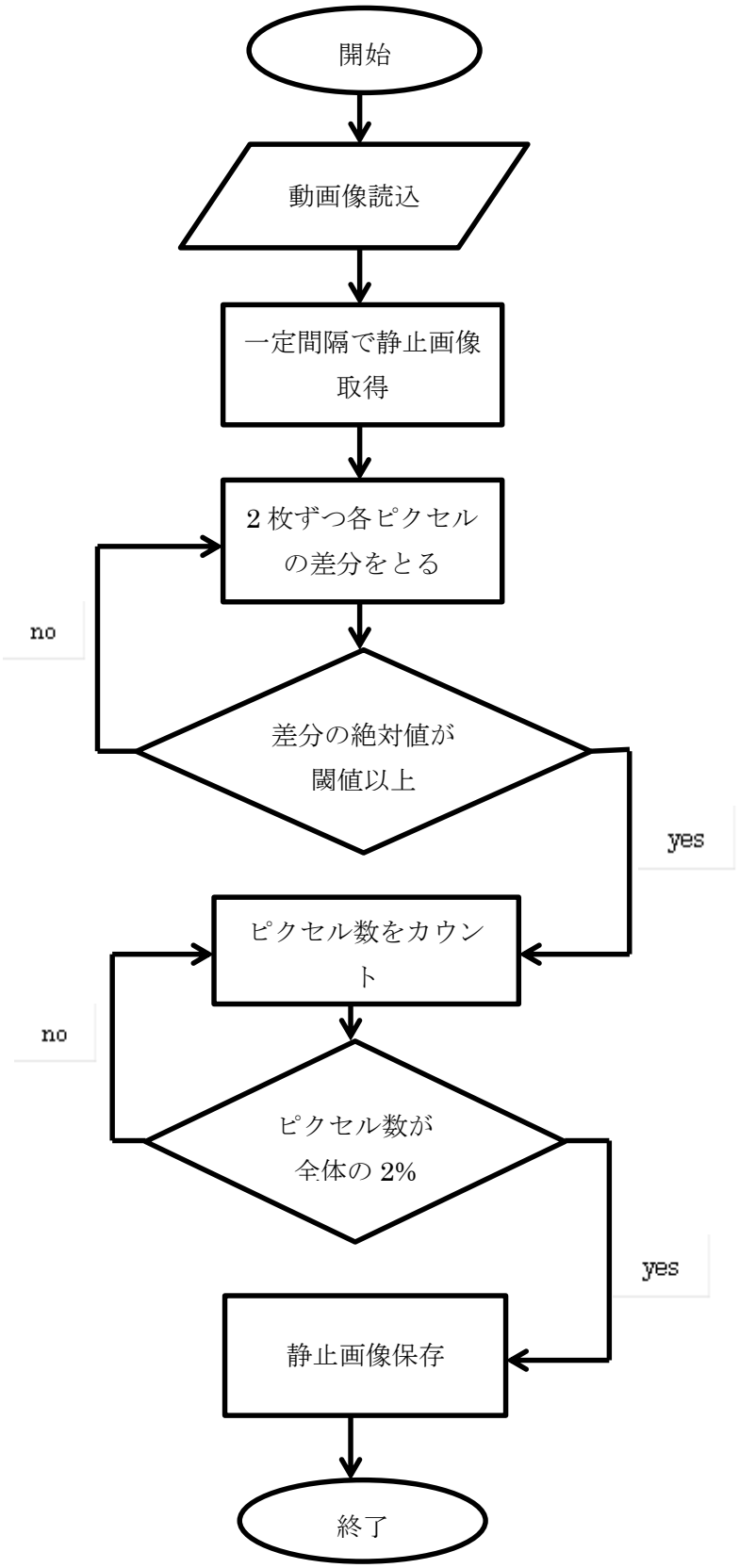

図 4.1：静止画抽出におけるフローチャート

4.3　背景差分による物体の検出

4.2 で保存された静止画像を 2 枚ずつ順番に背景差分をとる．例えば，保存された画像が 4 枚ならば 1 枚目と 2 枚目の静止画像の背景差分をとり，次に 3 枚目と 4 枚目の背景差分をとる．

静止画像を保存する時と同様に，2 枚の画像をピクセル毎に比較し差分を絶対値でとっていく．閾値を設定し，閾値以上の差分値になれば，比較した画像の色を取得し，背景差分後の画像を生成する．閾値以下の差分値になった場合は白色を取得することで背景が白色になり，マッチングを行った際に認識する確率をあげる．これにより，冷蔵庫に出し入れされた食品が静止画像となって取り出されると考えた．

4.4　物体のマッチング

4.3 でとり出された食品の静止画像をテンプレートに登録しておいた静止画像とマッチングしていく．

テンプレート画像と背景差分後の画像ではサイズが異なるため，背景差分後の画像の食品部分のみを輪郭抽出し，テンプレート画像と同じサイズに拡大する．傾きや向き等を考慮し，テンプレート画像を回転させながらマッチングしていくことでより認識率が上がると考えた．また，正面表のテンプレート画像だけでは不十分であると考え，横方向や裏面等，1 つの食品に対し，あらゆる方向でのテンプレート画像を用意しておき，どれか 1 つでもマッチすれば登録しておいた食品名が表示されるようにすることで，更なる認識率の向上が期待できると考えた．

マッチングに関しても 4.2，4.3 で述べたように 4.3 で生成された静止画像と用意したテンプレート画像を 1 枚ずつ比較していき，ピクセル毎の差分値を計算していく．各ピクセルの差分値が設定した閾値以上となった場合，ピクセルをカウントしていく．全てのピクセルを比較し，閾値以上となったピクセル数が画像全体の 50%となったとき，テンプレートとマッチしたと判断しテンプレート画像に登録しておいた食品名を表示させる．

図 4.2 と図 4.3 に背景差分及びテンプレートマッチングのフローチャートを示す．

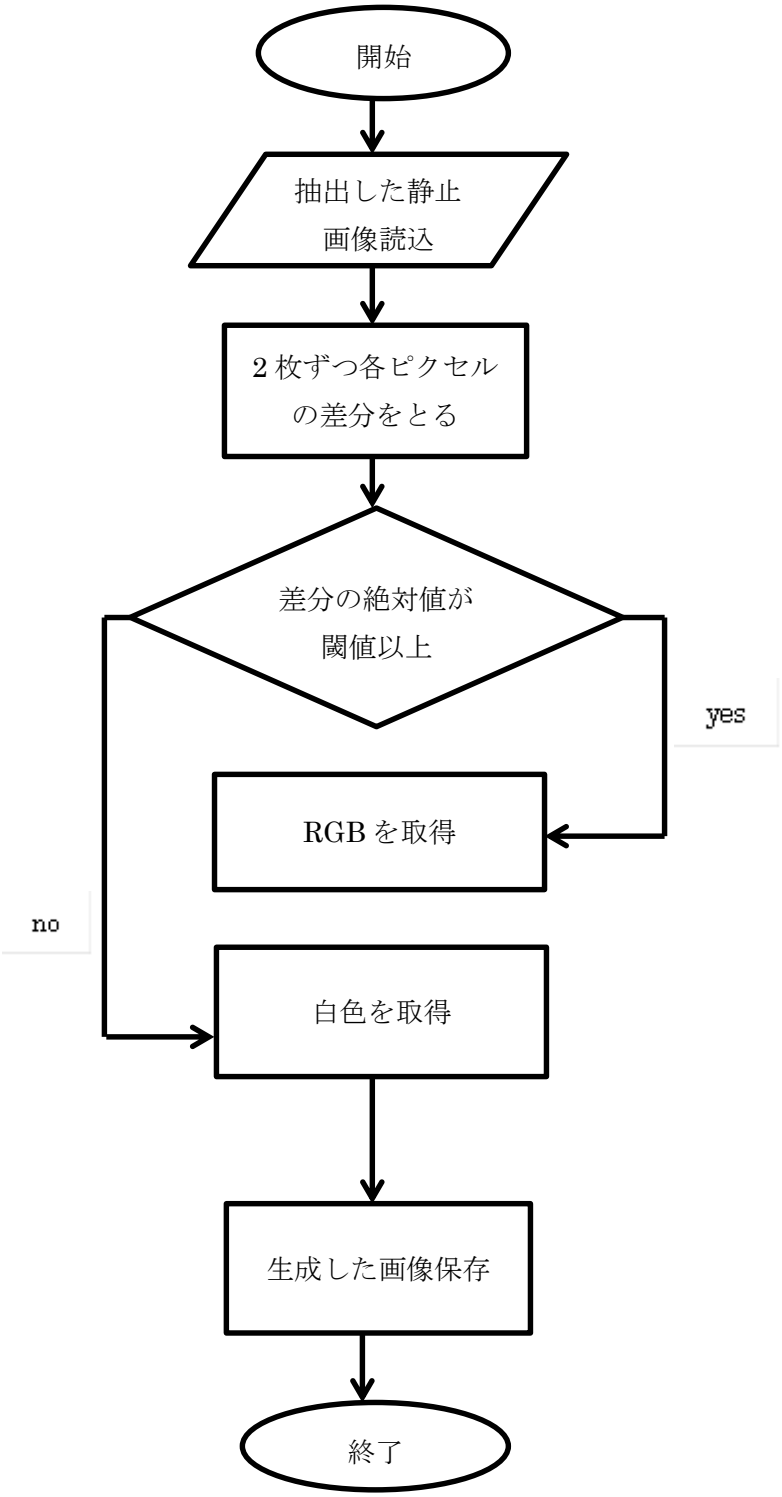

図 4.2：背景差分後の画像生成におけるフローチャート

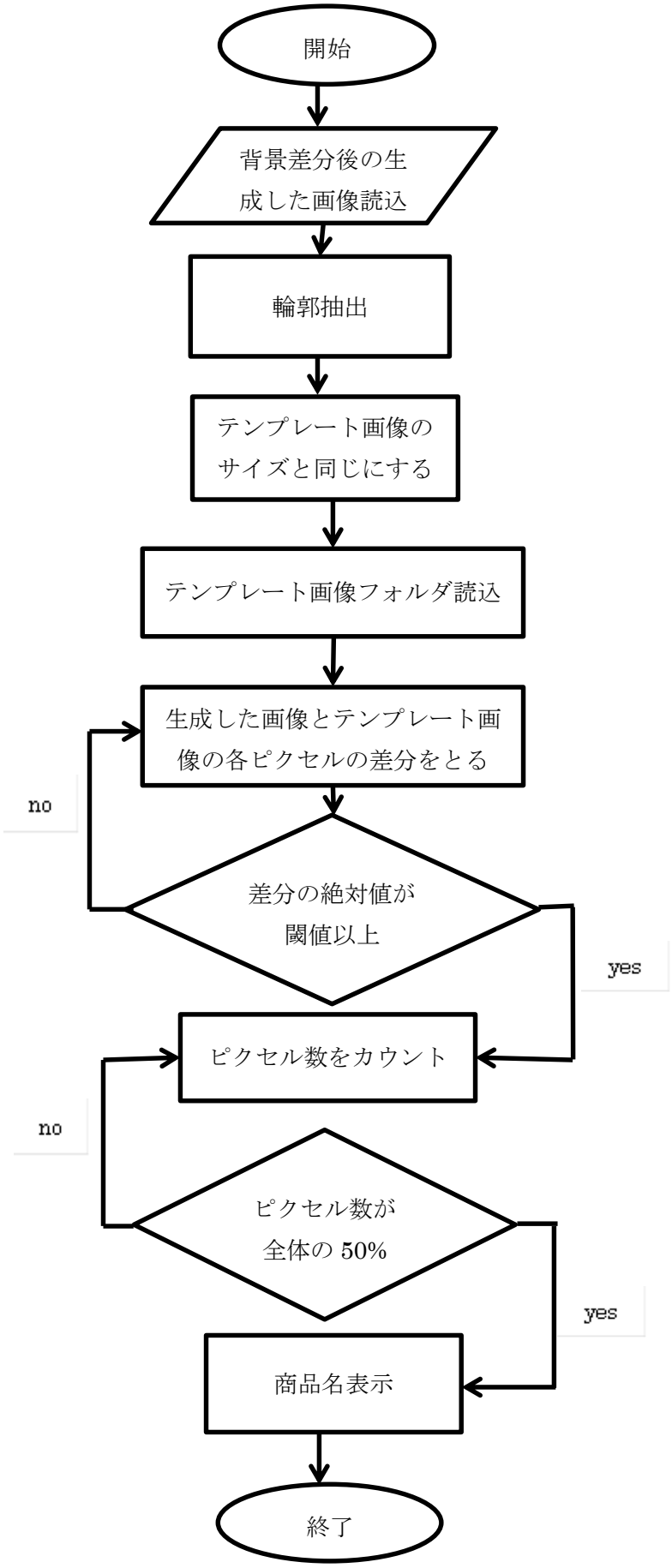

図 4.3：テンプレートマッチングにおけるフローチャート

第 5 章　実験・考察

5.1　動画像から自動で静止画像を抽出する実験

5.1.1　実験目的

本実験は，冷蔵庫内の動画像から 1 つずつ物を出し入れする直前と直後のフレームが自動で正確に抽出できるかを確認するために行った．

5.1.2　実験内容

本実験では，実際に冷蔵庫内を撮影する環境が用意できなかったため，研究室にある棚を冷蔵庫と仮定して実験を行った．棚の右上の位置に web カメラを固定して設置し，斜め上の位置から撮影した．棚のサイズは 860×315×430(mm)である．食品を 2 種類用意し，1 つずつ棚に入れた場合と予め 2 つ棚に食品をおいておき，2 つ取り出した場合の 2 パターンの動画を撮り，正確に静止画像を抽出できるか検証した．本実験で使用した食品名は，オロナミン C とつむぎのお茶である．棚に入れる場合は，オロナミン C，つむぎのお茶の順に入れ，取り出す場合はオロナミン C，つむぎのお茶の順に取り出した．図 5.1 に実験の様子を図で示す．また，各ピクセルの差分値の閾値は 200 と設定し，閾値以上となったピクセル数が画像全体の 2%以上となった場合に静止画像を保存するよう設定した．

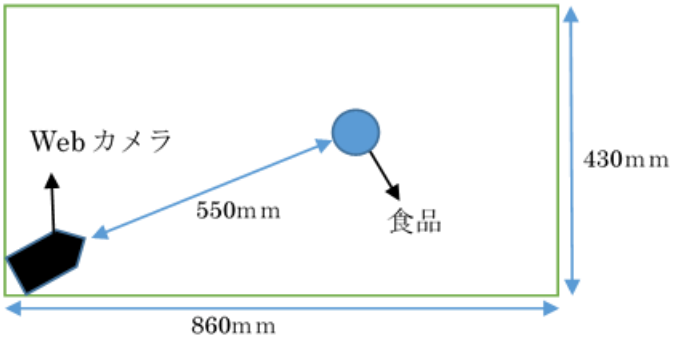

図 5.1：実験環境(上から見た図)

5.1.3　実験結果と考察

まず，食品を 1 つずつ棚に入れた場合の動画像から取得されたフレームを図 5.2~図 5.9 に示す.

図 5.2：フレーム 1

図 5.3：フレーム 2

図 5.4：フレーム 3

図 5.5：フレーム 4

図 5.6：フレーム 5

図 5.7：フレーム 6

図 5.8：フレーム 7

図 5.9：フレーム 8

以上の 8 枚のフレームが取得された．次に，取得されたフレームから抽出された静止画像を図 5.10～図 5.13 に示す.

図 5.10：抽出された静止画像 1

図 5.11：抽出された静止画像 2

図 5.12：抽出された静止画像 3

図 5.13：抽出された静止画像 4

図より，食品を入れる直前のフレームと入れた直後のフレームが抽出されていることがわかる．同様に食品を棚から取り出した場合の抽出されたフレームを図 5.14～図 5.17 に示す．

図 5.14：抽出された静止画像 1

図 5.15：抽出された静止画像 2

図 5.16：抽出された静止画像 3

図 5.17：抽出された静止画像 4

図より食品を取り出す直前と取り出した直後のフレームが抽出されていることがわかる．

以上より，食品を 2 つ順番に出し入れした場合，それぞれの食品を出し入れした前後の静止画像が抽出できた．これより，動画像から食品を出し入れする直前と直後の静止画像の自動抽出が可能であると考える．2 つだけでなく，3 つ 4 つといくつ出し入れしても上記の結果のように静止画像を抽出することが可能であるが，両手で一気に食品を出し入れするなど，食品を出し入れし，次の食品を出し入れする間，全く動きがない状態がない場合が生じるとうまく静止画像を抽出できないことがある．この問題点に関しては，今後の課題とする．また，閾値を変えて同様の実験を行ったところ，食品を出し入れしている最中のフレームが抽出されてしまい腕が映り込んだり，ブレ等が生じてしまった．閾値に関しては各ピクセルの差分値の閾値は 200 と設定し，閾値以上となったピクセル数が画像全体の 2%以上となった場合が適切であると言える．

5.2　抽出した静止画像から 1 つの食品を割り出す実験

5.2.1　実験目的

本実験は，抽出された静止画像の背景差分，及びテンプレートマッチングを通して，実際に食品を割り出すことが可能であるかを確かめるために行った．

5.2.2　実験内容

本実験では，5.1 で行った実験と同様，棚を冷蔵庫と見立てて実験を行った．棚の右上の位置に web カメラを固定して設置した．数種類の食品を用意し，1 つ棚に入れる度に撮影を行った(図 5.1)．

実験に使用した食品は飲料類，冷凍食品，果物，野菜等全部で 12 種類である．どの食品も一般家庭において比較的よく購入していると考えられる食品を用意した．用意した食品の画像を用意し，テンプレートとしてテンプレートフォルダに保存しておいた．テンプレートとして用いた画像はネットから探し，食品の正面全体が大きく表示されているものを使用した．

撮影した動画像をプログラムに通し実行することで，静止画像を抽出し，抽出された静止画像を背景差分及びテンプレートマッチングを行い，予めテンプレート画像に登録しておいた食品名が正確に認識し，表示されるか否かを確認した．また，背景差分における各ピクセルの差分値の閾値は 250 と設定し，テンプレートマッチングにおける各ピクセルの差分値の閾値は 300 と設定し，閾値以上となったピクセル数が画像全体の 80%以上となった場合，テンプレートに登録した食品名が表示されるように設定した．

5.2.3　実験結果と考察

撮影した食品がどのように認識されたかを例として以下に記す．

・撮影した食品：チョコレート(お土産)
・登録したテンプレート画像の食品名：チョコレート
・抽出された静止画像(図 5.18~図 5.19)

図 5.18：チョコレートを入れる直前

図 5.19：チョコレートを入れた直後

・背景差分

抽出された静止画像による背景差分で得られた結果を図 5.20 に示す．

図 5.19 と図 5.20 を比べると少し影や色が薄くなっている部分はあるが，文字やパッケージの模様等は鮮明に見えることがわかる．背景を白くすることでよりはっきりと食品が抽出されている．これより背景差分は成功したといえる．

図 5.20：背景差分によって抽出された画像

・テンプレートマッチング

背景差分によって抽出された画像を拡大し，テンプレートフォルダに登録しておいた 12 種類の食品とそれぞれテンプレートマッチングを行った時のそれぞれのマッチング率を表 5.1 に示す．

表 5.1：マッチング率

テンプレート	抽出画像とのマッチング率	認識結果
1000ml パック(牛乳)	3.2%	失敗
500ml パック(カフェオレ)	38.4%	失敗
バター	42.7%	失敗
たまご(10 ケ入り)	4.1%	失敗
300ml 缶(ビール)	5.1%	失敗
500ml ペットボトル(午後の紅茶)	37.5%	失敗
トマト	2.3%	失敗
りんご	5.2%	失敗
冷凍食品(えびピラフ)	57.1%	失敗
カップ類(焼プリン)	62.9%	失敗
トレイ類(豚ひき肉)	11.3%	失敗
チョコレート	81.6%	成功

表 5.1 より，チョコレートが 81.6%で認識されたため，チョコレートと表示された．よって成功したと言える．同様に他の食品で行った場合の認識結果を表 5.2 に示す．

表 5.2：食品の認識結果

食品名	認識結果	備考
1000ml パック(牛乳)	成功	マッチング率 85.2%
500ml パック(カフェオレ)	失敗	認識はしたが焼プリンも認識
バター	成功	マッチング率 84.1%
たまご(10 ケ入り)	成功	マッチング率 86.9%
300ml 缶(ビール)	成功	マッチング率 85.7%
500ml ペットボトル(午後の紅茶)	成功	マッチング率 88.8%
トマト	失敗	認識はしたがトマトも認識
りんご	失敗	認識はしたがりんごも認識
冷凍食品(えびピラフ)	成功	マッチング率 81.3%
カップ類(焼プリン)	成功	マッチング率 82.2%
トレイ類(豚ひき肉)	失敗	マッチング率 66.2%

以上の結果より形や柄が全く違う食品に関しては認識成功した．豚ひき肉等のトレイ類は上からのアングルで撮影し，同じお店の豚ひき肉をテンプレート画像として登録しておくことで，トレイに貼られているバーコードシールや文字等を読み取りマッチングが成功したが，似た色と形をした牛ひき肉も認識してしまった．他に，カフェオレの認識時に焼プリンも認識してしまった．このように似たパッケージや色の食品が同じように認識されてしまい正確なデータが得られないという問題点が挙げられる．これを解決するためには食品のわかりやすい部分にバーコードシール等を予め貼っておくことでそのバーコードが動画で撮影されればテンプレートマッチングより優先的に認識し，食品を正確に特定できるのではないかと考える．また，トマトとりんごに関しても似た色と形であるため，どちらも認識してしまった．野菜等のシール等を貼れない食品に関しては，マッチングの閾値をあげることで解決できるのではないかと考えたが，閾値を上げ過ぎるとうまく認識できない場合があるため，自動で適度な閾値を設定できれば識別可能になるのではないかと考えた．

第 6 章　結論

本研究では，冷蔵庫内の食品を音声認識や手動ではなく，自動で管理する方法について考察した．自動管理に向けて，冷蔵庫内に web カメラを設置し，動画を撮影し，動画像を解析する手法を提案した．実験結果から，動画像内の静止画像抽出とテンプレートマッチングによる食品の特定は成功した．web カメラの位置を調整する，あるいは数を増やすことでドアポケットや野菜室の食品及び食材も認識可能であると考える．しかし，肉や魚などトレイに入った食品やトマトやりんご等の色やパッケージが似ている食品に関しては間違った食品も認識してしまうため，第 2 章の 2.1 で述べたユーザとの音声対話による補正や，専用のバーコードシールを貼る等といった策が必要であると考える．他にも一度に複数の食品を同時に出し入れした場合は正確な認識ができなかったため，今後の課題である．また，本実験では，食品によって背景差分やマッチングの閾値を手動で変更しているため，今後は適切な閾値を自動で設定できるようにしていきたい．

謝辞

本研究を進めるにあたり，ご指導を頂いた卒業論文指導教員の三好力教授に感謝いたします．また，日常の議論を通して多くの知識や示唆を頂いた三好研究室の皆様，友人，知人の皆様 に感謝いたします．

参考文献

[1]松本拓也，白井良明，島田伸敬，”画像処理と音声対話による冷蔵庫内の食品管理システム”，情報処理学会第 71 回全国大会，2-419-420
[2]加茂田玲奈，上田真由美，船冨卓哉，飯山将晃，美濃導彦，”食材管理のための荷重特徴を用いた食材同定”，社団法人，電子情報通信学会，181-186(2012)
[3] 冷蔵庫・食材を管理 おすすめアプリランキング
URL　http://app-liv.jp/foods/cooking/1654/
[4]ともだち家電:シャープ
URL　http://www.sharp.co.jp/tomodachi/
[5]OpenCV.jp
URL　http://opencv.jp/
[6] OpenCV での画素へのアクセス方法
URL　https://miyabiarts.wordpress.com/
[7]OpenCV で背景差分
URL　http://whoopsidaisies.hatenablog.com/
[8]OpenCV のテンプレートマッチング
URL　https://blog.daionet.gr.jp/knok/2015/02/27/opencv-template-matching/
[9]西友-ネットスーパー
URL　https://www.the-seiyu.com/front/contents/top/

付録

```
 #include<opencv2/opencv.hpp>
#include <iostream>
#include<stdio.h>

using namespace cv;
using namespace std;

void backdiff(char* filename1, char* filename2); //背景差分の関数
void matching(char* filename1, char* filename2, int itemno); //マッチングの関数(改良の余地あり　回転等)
int gettemplateinfo(void); //テンプレート、商品情報の取得(現在はファイルから)

int dno; //背景差分ファイル名の番号
char templatename[10][50]; //マッチング用テンプレート名[商品の登録数][名前の長さ]
char itemname[10][200]; //商品名
int itemcount; //実際に登録されている商品数

int main(int argc, char* argv[])
{
  int x, y;
  int b,g,r;
  int B,G,R,P;
  int count;   //カウント用変数
  int flg=2;
  uchar p1[3], p2[3];
  IplImage *img;
  IplImage *img1;
  IplImage *img2;
  IplImage *img3;
  CvCapture *cap;
  char filename[50];
  char filename1[50];
  char filename2[50];
  int cnt,no;
  int i,j;
  int fno=0;
  char filenamef[50];
  char filenames[50];
  char filenamee[50];
  char filenamed[50];
  char filenamet[50];

  dno=0;

  if(gettemplateinfo()!=0){ //テンプレート、商品情報の取得
    return 0;
  }

  cap = cvCaptureFromFile("in2.webm");

  //ここで動画からあるフレーム間隔で静止画をとる
  cnt=1;
  no=0;
  for(i=0;i<150;i++){
    img = cvQueryFrame(cap);
    if(img==NULL) break;
    if((cnt-1) % 5 == 0){
      sprintf(filename, "img%d.bmp",++no);
      cvSaveImage(filename,img);
    }
    cnt++;
  }

  //ここから静止画を比較する
  for(i=0;i<no-1;i++){
    sprintf(filename1,"img%d.bmp",i+1);
    sprintf(filename2,"img%d.bmp",i+2);
    img1 = cvLoadImage (filename1, CV_LOAD_IMAGE_COLOR);
    img2 = cvLoadImage (filename2, CV_LOAD_IMAGE_COLOR);
    img3 = cvCreateImage (cvSize(640,480),IPL_DEPTH_8U,1);

    count=0;
    for (y = 0 ; y < img1->height; y++) {
      for (x = 0 ; x < img1->width; x++) {

    //一枚目の画素値
    p1[0] = img1->imageData[img1->widthStep * y + x * 3]; // B
    p1[1] = img1->imageData[img1->widthStep * y + x * 3 + 1]; // G
    p1[2] = img1->imageData[img1->widthStep * y + x * 3 + 2]; // R

    //二枚目の画素値
    p2[0] = img2->imageData[img2->widthStep * y + x * 3]; // B
    p2[1] = img2->imageData[img2->widthStep * y + x * 3 + 1]; // G
    p2[2] = img2->imageData[img2->widthStep * y + x * 3 + 2]; // R

    //1 ピクセル B,G,R それぞれの差分
    b = p1[0] - p2[0];
    g = p1[1] - p2[1];
    r = p1[2] - p2[2];

    //絶対値に変換
    B = abs(b);
    G = abs(g);
    R = abs(r);

    //1 ピクセルの合計差分
    P = B + G + R;

    //しきい値以上となったピクセル数
    if(P>=150) {
      count++;
    }
      }
    }
    if(count>=img1->height*img1->width*0.03){ //しきい値以上となったピクセル数が画像全体の 2%以上となった場合
      if(flg == 1){ //動きが発生した直前の画像を保存する
    fno++;
    sprintf(filenamef, "img_f%d.bmp",fno);
    cvSaveImage(filenamef,img1);
      }
      flg=0;
    }
    else{
      if(flg<1){ //動きが止まった直後の画像を保存する
    fno++;
    sprintf(filenamef, "img_f%d.bmp",fno);
    cvSaveImage(filenamef,img2);
      }
      flg=1;
    }
  }

  for(i=0;i<fno;i+=2){
    //商品の出し入れが行われた時の背景差分をとる
    sprintf(filenames,"img_f%d.bmp",i+1);
    sprintf(filenamee,"img_f%d.bmp",i+2);
    backdiff(filenames,filenamee);

    //背景差分で得られた商品の特定をテンプレートマッチングで行う
    sprintf(filenamed, "img_d%d.bmp",(int)(i/2)+1);
    for(j=0;j<itemcount;j++){
```

```
      strcpy(filenamet,"template/");
      strcat(filenamet, templatename[j]);
      matching(filenamed,filenamet, j);
    }
  }
  printf("END\n");
}



void backdiff(char* filename1, char* filename2) //背景差分
{
  int x, y;
  int b,g,r;
  int B,G,R,P;
  int count;   //カウント用変数
  uchar p1[3], p2[3];
  IplImage *img1;
  IplImage *img2;
  IplImage *img3;
  char filenamed[50];

    img1 = cvLoadImage (filename1, CV_LOAD_IMAGE_COLOR);
    img2 = cvLoadImage (filename2, CV_LOAD_IMAGE_COLOR);
    img3 = cvCreateImage (cvSize(640,480),IPL_DEPTH_8U,3);

    count=0;
    for (y = 0 ; y < img1->height; y++) {
      for (x = 0 ; x < img1->width; x++) {

    //一枚目の画素値
    p1[0] = img1->imageData[img1->widthStep * y + x * 3]; // B
    p1[1] = img1->imageData[img1->widthStep * y + x * 3 + 1]; // G
    p1[2] = img1->imageData[img1->widthStep * y + x * 3 + 2]; // R

    //二枚目の画素値
    p2[0] = img2->imageData[img2->widthStep * y + x * 3]; // B
    p2[1] = img2->imageData[img2->widthStep * y + x * 3 + 1]; // G
    p2[2] = img2->imageData[img2->widthStep * y + x * 3 + 2]; // R

    //1 ピクセル B,G,R それぞれの差分
    b = p1[0] - p2[0];
    g = p1[1] - p2[1];
    r = p1[2] - p2[2];

    //絶対値に変換
    B = abs(b);
    G = abs(g);
    R = abs(r);

    //1 ピクセルの合計差分
    P = B + G + R;

    if(P>=80) { //しきい値以上となったピクセル値を BGR で取得する
      img3->imageData[img3->widthStep * y + x * 3] = img2->imageData[img2->widthStep * y + x * 3];
      img3->imageData[img3->widthStep * y + x * 3 + 1] = img2->imageData[img2->widthStep * y + x * 3 + 1];
      img3->imageData[img3->widthStep * y + x * 3 + 2] = img2->imageData[img2->widthStep * y + x * 3 + 2];
    }
    else{ //しきい値未満の場合は白色とする
      img3->imageData[img3->widthStep * y + x * 3] = 255;
      img3->imageData[img3->widthStep * y + x * 3 + 1] = 255;
      img3->imageData[img3->widthStep * y + x * 3 + 2] = 255;
    }
      }
    }
    dno++;
    sprintf(filenamed,"img_d%d.bmp",dno); //背景差分のファイル名
    cvSaveImage(filenamed,img3); //背景差分のファイル保存
    return;
}

void matching(char* filename1, char* filename2, int itemno) //マッチング
{
  int x, y;
  int x1,y1;
  int b,g,r;
  int B,G,R,P;
  int count;   //カウント用変数
  uchar p1[3], p2[3];
  IplImage *img;
  IplImage *img1;
  IplImage *img2;

    img1 = cvLoadImage (filename1, CV_LOAD_IMAGE_COLOR);
    img2 = cvLoadImage (filename2, CV_LOAD_IMAGE_COLOR);

    for (y = 0 ; y < img1->height-img2->height; y++) { //高さ方向で 1 ピクセルずつずらしていく
      for (x = 0 ; x < img1->width-img2->width; x++) { //幅方向で 1 ピクセルずつずらしていく
    count=0;
    for (y1 = 0 ; y1 < img2->height; y1++) { //テンプレートの高さ分ピクセル値をチェックする
      for (x1 = 0 ; x1 < img2->width; x1++) { //テンプレートの幅分ピクセル値をチェックする

    //一枚目の画素値
      p1[0] = img1->imageData[img1->widthStep * (y+y1) + (x+x1) * 3]; // B
      p1[1] = img1->imageData[img1->widthStep * (y+y1) + (x+x1) * 3 + 1]; // G
      p1[2] = img1->imageData[img1->widthStep * (y+y1) + (x+x1) * 3 + 2]; // R

    //二枚目の画素値
    p2[0] = img2->imageData[img2->widthStep * y1 + x1 * 3]; // B
    p2[1] = img2->imageData[img2->widthStep * y1 + x1 * 3 + 1]; // G
    p2[2] = img2->imageData[img2->widthStep * y1 + x1 * 3 + 2]; // R

    //1 ピクセル B,G,R それぞれの差分
    b = p1[0] - p2[0];
    g = p1[1] - p2[1];
    r = p1[2] - p2[2];

    //絶対値に変換
    B = abs(b);
    G = abs(g);
    R = abs(r);

    //1 ピクセルの合計差分
    P = B + G + R;

    //しきい値以上となったピクセル数
    if(P<=300) {
      count++;
```

```
    }
     }
    }
    if(count>(img2->height*img2->width*0.80)){
 //しきい値以上となったピクセル数が画像全体の○%以
上となった場合背景差分で得た商品とテンプレート画
像が一致したとみなしその商品名を表示する
        printf("商品名 = %s\n",itemname[itemno]);
      //背景差分画像とテンプレート画像で一致とみなし
た位置を表示する
      // CvPoint p1=cvPoint(x,y);
      //CvPoint p2=cvPoint(x+img2->width,y+img2-
>height);
      //cvRectangle(img1,p1,p2,CV_RGB(255,0,0),2,CV_
AA,0);
      // cvNamedWin-
dow("img1",CV_WINDOW_AUTOSIZE);
      // cvShowImage("img1",img1);
      //cvWaitKey(0);
      //cvReleaseImage(&img1);
      //cvDestroyWindow("img1");
      return;
    }
     }
    }
    return;
}

int gettemplateinfo(void){ //テンプレート、商品情報の
取得(現在はファイルから)
  FILE*fp;
  char filename[100];
  int ret;

  //テンプレートフォルダにあるテンプレート、商品情
報ファイルから
  //テンプレート画像のファイル名と商品名を取得する
  strcpy(filename,"template/template.csv");
  fp=fopen(filename,"r");
  if(fp==NULL){
    printf("template.csv ファイルが見つかりません。
\n");
    return -1;
  }

  itemcount=0;
  while(fscanf(fp,"%s %s",templatename[itemcount],it
emname[itemcount])!=EOF){
    itemcount++;
  }

  fclose(fp);
  return 0;
}
```