平成27年度 修士研究論文

食事画像からの
自動カロリー量推定システムの実現

電気通信大学 大学院情報理工学研究科
総合情報学専攻 メディア情報学コース
1430014　　岡元 晃一
主任指導教員　柳井 啓司 教授
指導教員　　高橋 裕樹 准教授

平成28年1月29日

概 要

近年では, 健康的至高の高まりにより食事記録を付ける人が増えてきている. それに伴い食事記録支援システムが多く公開され始めているが, 既存のシステムのほとんどが正確にカロリー量を推定することができない. そこで本論文では食品を基準物体と撮影することで, 食品の認識を行い, さらに大きさを推定することでその食品のカロリー量を推定するシステムを提案する. システムはユーザーの携帯性や利便性を考えスマートフォンアプリという形での実装を行う. システムは画像中より食品領域及び基準物体領域を抽出し, その大きさを比較する. 基準物体は事前に面積がわかっていること以外には制約はなく, ユーザーが各々常に携帯しているものを使用することが出来る. 食品認識部分の手法には高精度な認識が可能なディープラーニングを用いた. 一般にディープラーニングによる画像認識は計算量が多くモバイルでの利用は難しいが, パラメータ数が少ないネットワークを選択したりなどの工夫により, サーバを介さずモバイル上での実行ながら約 0.2 秒程度での実行速度で高精度な認識を可能にした.

実験ではカロリー量推定実験とユーザー評価実験の 2 つを行い結果としてカロリー量推定実験での誤差の平均は 52.231kcal, 相対誤差の平均は 0.213 となった. ユーザー評価実験でも既存システムよりも記録を取りやすいという評価を得た. このことから提案システムの有効性が確認できた.

目 次

第1章

はじめに

1.1　背景

近年, 健康的思考の高まりにより食事記録を付ける人が増えてきている. しかしそれには写真を撮影したり, 食べ物の名前を入力したり, カロリー計算を行わなければなかったりと非常に手間がかかり, 記録を付けること億劫になり記録を付けることが続かない恐れがある.

先行研究として河野ら [1] や我々[2] が食事記録支援システムを開発しているが, これらはユーザーの主観によってカロリー量を変更したり, 事前に決められたカロリー量を食べた回数と積算し総カロリー量を求めたりと簡易的な方法でカロリー量推定を行っているため, 必ずしも正確なカロリー量を推定できるわけではなかった. 特に人の主観的な大きさ, カロリー量評価では, 栄養士などの方を除く, 栄養などの知識が無い人間が行うと, 正しく評価を行うことが難しい. また現在では食事の写真を撮影し, サーバにアップロードすることで栄養士の方にカロリー量などの情報をチェックしてもらえるようなアプリ [3] も公開されているが, 定額の金額を払わなくてはならなかったりとコストがかかり, 手軽に使用できるものではない.

そこで本研究では, 基準物体と食品を同時に撮影することで事前に食品や栄養などの知識が無いユーザーでも, 食品の名前はもちろん, 大きさに応じたカロリー量を自動で推定するシステムの作成を目指す.

1.2　目的

本研究では食品と事前に登録しておいた基準物体とが一緒に写った画像を撮影することでその画像から食べ物の名前や面積, カロリー量を取得することを目的と

する. システムはユーザーの携帯性や利便性を考え, スマートフォンアプリでの実装を行う. 使用する基準物体とは, 事前に面積がわかっていること以外には制約はなく, ユーザーが各々常に携帯しているものを使用することができる.

カロリー量推定では, 画像中の食品及び基準物体の領域を抽出し, 事前に大きさのわかっている基準物体領域と食品領域を比較することで食品領域の面積を求め, その面積に応じたカロリー量を推定する. また本論文で, “食事” とは複数の食品が組み合わさり一度に摂取するもの全体を示し, “食品” とは調理された一つの料理など (ラーメンやごはんなど) 食事を構成するものの 1 つとする. システムの使用風景を図 1.1, キャプチャ画面を図 1.2 に示す.

図 1.1: 提案システムの使用風景

図 1.2: 提案システムのキャプチャ画面

1.3　本論文の構成

本論文の構成は以下のようになっている.

1 章 はじめに
　この研究の背景, 目的を記載する.
2 章 関連研究
　関連研究を紹介する.
3 章 システムの概要
　今回作成するシステムの概要について記述する.
4 章 システムの詳細
　今回作成するシステムの詳細について記述する.
5 章 作成したシステム
　作成したシステムについて記述する.
6 章 実験
　実験について記述する.
7 章 考察
　実験結果より考察を行う.
8 章 終わりに
　まとめと今後の課題について記述する.

第2章

関連研究

2.1　食事記録支援システム

現在食事記録支援システムといえば, 河野ら [1] のシステムや Foodlog[4] などが挙げられるが, これらは認識された食べ物の基準的なカロリー量だけを返したり, ユーザーの手によってカロリー量を決定するので真にユーザーが食べたカロリー量を返すようなシステムではない.

Foodlog では撮影した画像をサーバにアップロードし, 画像認識を行う. しかしこのときの結果はおおよその種類, カロリー量を返すだけである.

河野らはスマートフォン上でリアルタイムに食事画像認識を行うことによって, 食品名はもちろん, カロリー量においてもユーザーの主観によって可変させ記録するシステムを作成した. しかし, これはあくまでユーザーの主観によって決まってしまうので, ユーザーによっては多く見積もってしまったり, 反対に少なく見積もってしまったりなど, 正確なカロリー量を推定することはできない.

また Wu ら [5] は対象をファストフードに絞ることで, 画像認識を行いインターネット上に記載されているカロリー量情報を記録することで食事記録を付けることが出来るシステムを提案している. しかしこれには 4 点からの画像が必要だったり, 動画を撮影しなければならなかったりする手間がかかる. またこれはインターネット上にカロリー量などが記載されているファストフードなどに関しては有効だが, 他の食品には適用しづらい問題がある.

そこで本システムではユーザーにかかわらず正確なカロリー量を推定しかつより手軽で, 様々な食品に対応できるようなシステムを提案する.

2.2　食事調査法

カロリー量の推定は食事記録につながり, 栄養学の分野で行われてる食事調査法の手助けになるとも考えられる. 現在食事調査法で主流なのものは 24 時間思い出し法や, 食物摂取頻度調査法, 陰膳法などが挙げられる [6]. 24 時間思い出し法や食物摂取頻度調査法は手軽に行えるというメリットがあり, 広く使われている. しかし, これは 1 日に食べたものを全て覚えておかなくてはならず調査対象者への負担が大きい. なおかつ負担が大きい割には食事記録の精度もあまり良くないとされている.

逆に陰膳法は実際に摂った食事を 1 人前多く作ってもらいそれを基にしたり, 実際に栄養士に再現してもらった食事から栄養素を記録する方法である. 実際の食事を基にするので食事記録の精度は高いが, 余分に食事を作らなくてはならないので人手と金額との両方のコストがかかるというデメリットがある.

そこで今回は画像を 1 枚取るだけと調査対象者の負担の少なく, あまりコストのかからないシステムを提案する.

2.3　食事領域分割

下田ら [7] は CNN を用いて複数の食品が写っている食事画像から, それぞれの食品領域ごとにセグメンテーションを行っており, さらに食品のサイズ推定を行っている. この時には味噌汁はどの場面においても大きさがあまり変わらないと仮定し, そのサイズから他の食品のサイズを求めている. このような精度の良い手法を用いることも考えたが, この手法は非常に計算量が多くスマートフォンなどのモバイルでの実行は難しい.

また Joachim ら [8] は机上の皿の検出を行い, その領域内から食品のセグメンテーションも行っている. 皿の検出は 99%以上の精度を示しており, 食品のセグメンテーションにおいても 90%近い精度を示した. しかし現状では PC 上での実行しか行っておらず, モバイルでの実行速度などに疑問がある.

そこで, 今回は複数品においてもモバイルという特性を活かし, 動かしながら撮影を行い, 対象食品ごとを大きく映すことで余計なセグメンテーションの手間を省くようにし, 皿の検出にも独自にエッジ検出を用いた手法を採用する.

2.4　手動カロリー量推定システム

手動でのカロリー量推定システムはすでにアプリ化がされており, 代表的なものとして株式会社クオリアが公開している “カロナビ”[3] などが挙げられる. このシステムは食事の写真を撮影し, その写真をサーバへ送信する. その画像を栄養士の方がチェックすることでその食事のカロリー量や栄養素などを記録することが出来る. しかしこのシステムでは人手でチェックを行っているので返答までに時間がかかったり, 月々に定額の料金を払わなくてはならなかったりと気軽に使えるものではない.

そこで本システムでは画像認識を用いて, 人手を介さず時間とコストを減らした自動カロリー量推定システムを提案する.

2.5　自動カロリー量推定システム

自動カロリー量推定システムには現在様々なシステムが提案されている.

我々の提案した GrillCam[2] では食事シーンを撮影し, 口と箸を検出することでその 2 点が近づいた時を食べたと認識する (図 2.1). そして食べた瞬間の画像を切り出し, 画像認識を行い, 何を食べたのかを種類とカロリー量を記録するシステムである. 食事シーンを動画に撮影しているので, 食べる前に食べる量のわからない様な焼き肉や鍋といった料理にも適用可能で, どれだけ食べたかリアルタイムで総カロリー量を示すことが出来る. リアルタイムでのカロリー量を表示するので食べ過ぎの抑制にもなる. しかしカロリー量は食品ごとに一口大の決められた量を食べた回数と積算し, 求めているので真に正確なものとは言えない.

宮崎ら [9] は撮影した画像と “FoodLog” にある画像の類似度を検索し, その複数の画像のカロリー量から近いカロリー量を導き, 撮影した画像のカロリー量を推定する. しかしこれはあくまで画像として近いものを探しているだけであり, 面積や体積に応じたカロリー量を返すものではない.

Pouladzadeh ら [10] の研究では食品とユーザーの親指を同時に撮影することで指の大きさと比較を行い食品の大きさを求め, カロリー量を推定するシステムを提案している. しかし指の出し方や角度, 映り方などによっては食品の大きさに誤差が生じてしまう可能性がある.

Kong ら [11] は普段の食事を複数視点から撮影を行うことで, 食事の 3D 復元を行い体積よりカロリー量を推定する. さらにパッケージに記載されているカロリー量や栄養を文字認識を行い記録できると共に, 食事前と食事後の写真を撮影しどれ

だけの量を食べたのか, 残したのかというところに関しても記録できる. しかし事前にスマートフォンについてカメラの較正を行わなくてはならなかったり, 正確に較正した地点から撮影を行わなくてはならずユーザーに対する負担が大きい.

Xu ら [12] は複数枚, もしくは 1 枚の画像からカロリー量を推定するシステムを提案している. これには事前に大量の異なる視点から画像を撮影し, 記録しておかなくてはならない. それには非常に手間がかかり, データセットの拡張が非常に難しい.

Chen ら [13] は深度情報の付いている画像を用いることで認識及び量の推定を行い, カロリー量を推定している. しかし画像の深度情報は通常のスマートフォンなどでは取得することが難しくこのシステムはあまりモバイル向きではない.

Myers ら [14] はモバイルで動作し, 1 枚の画像でのカロリー量推定システムを提案している. これは画像より DeepLearning を用いて, 画像のピクセルの深度を測り, そこから食品の体積を求めカロリー量を推定するシステムである. しかし 2016 年 1 月時点ではアイデアは完成しているようだが, 実際の実験ではまだまだ実用段階にはほど遠く, 実装時のアプリも食品認識のみと論文に記述されている.

そこで本システムでは, 撮影時の誤差を少なくするためユーザーが常に携帯している名刺サイズのカードや財布などを基準物体として登録を行い, 食品と基準物体を真上から撮影することで誤差を可能な限り小さくする方法を選択する. 画像を 1 枚撮影するだけで良いので手間も大きくかからず, データセットの拡張においても食品のサイズごとに 1 枚のみ必要なので拡張も容易であると考えられる.

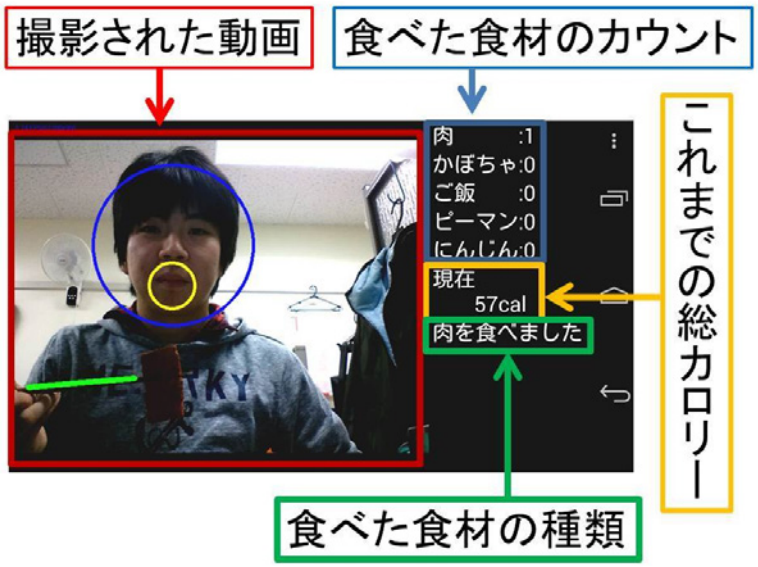

図 2.1: 自動カロリー量推定システムの例 ([2] より引用)

第3章

システムの概要

提案システムでは以下の流れでカロリー量の推定を行う.

1. 対象の食品と基準物体を一緒に撮影
2. 食品領域及び基準物体領域をそれぞれ抽出
3. 食品領域を用いて DCNN での画像認識
4. 食品領域と基準物体領域の大きさを比較
5. 求められた大きさよりカロリー量を推定

真上以外から撮影した場合，一般に射影歪みが生じて補正が必要になるが, 提案システムでは補正を行わずに済ますために, 食事画像をテーブル面に垂直に真上から撮影することを仮定する. 基準物体は面積がわかっており, 食品の左側に写っているということ以外には制約はなく, どのような大きさ, 形状のものでも使用することが出来る. 食品領域及び基準物体の抽出にはエッジ検出を用いて対象範囲を縮小してから, GrabCut を使用することで正確な領域抽出を実現している. さらに食品領域の検出には色情報を k-Means で分割することで食器上からさらに正確な食品領域を抽出している. 画像認識部分ではディープラーニングを使用している. パラメータ数の少ないネットワークの採用や, 様々な工夫でモバイル上での実行を可能にし, 実行速度も約 0.2 秒程度と非常に高速かつ精度の高いシステムとなっている. 実際のシステムで使用する画像を図 3.1 に, システムのフローチャートを以下の図 3.2 に示す.

図 3.1: システムで使用する画像例

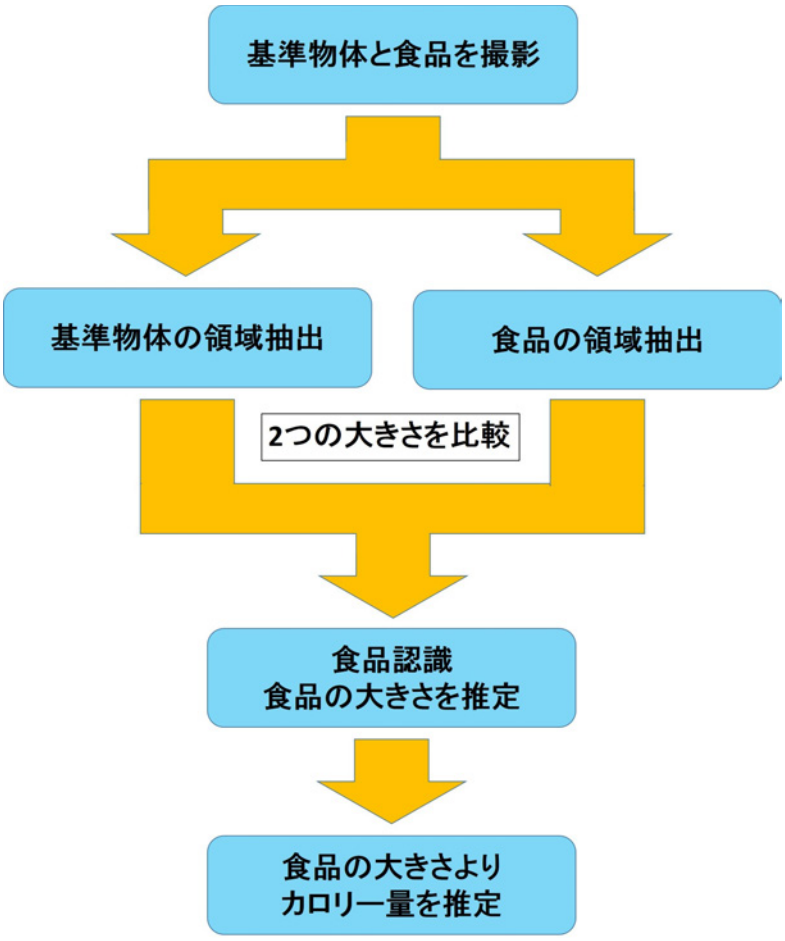

図 3.2: システムのフローチャート

第4章

システムの詳細

本章では, 本システムの詳細について記述する. 始めに食品領域抽出部分について, 次に基準物体領域抽出, 認識エンジンについて記述する.

4.1　食品領域抽出

ここでは食品領域抽出について述べる. 食品領域抽出は始めに食器領域を探索し, その範囲内から食品領域を探索する. そして狭められた範囲内で GrabCut を用いて食品領域を抽出するので精度良い抽出を可能にしている. 食品領域抽出のフローチャートを図 4.1 に示し, 以下にそれぞれの詳細を記述する.

4.1.1　食器領域検出

単純に画像全体から, グラフベースの領域分割手法である GrabCut による分割を行っても背景が大きく入ってしまい, 使用することができない. そこで始めに背景と食器領域の分割を行い背景が入ってしまう問題を解決する. これにはエッジ検出を用いて精度を高めている. 与えられた食品が入っているとされる領域についてエッジ検出を行うことで, 食器の輪郭を抽出することが出来る. そしてその部分だけを矩形として抽出を行い, その部分で GrabCut を行うことで大きく背景が入ってしまうという問題を解決した. エッジ検出を用いて食器矩形を求めた例が図 4.2, 食器矩形を用いて GrabCut を行った場合とそうでない場合の比較を以下の図 4.3 に示す.

しかしこれでもあくまで食器領域を抽出しただけに過ぎず, 食器の中に多く盛られているのか, 少しだけ盛られているのかがわからない. そこでこの矩形を用いて

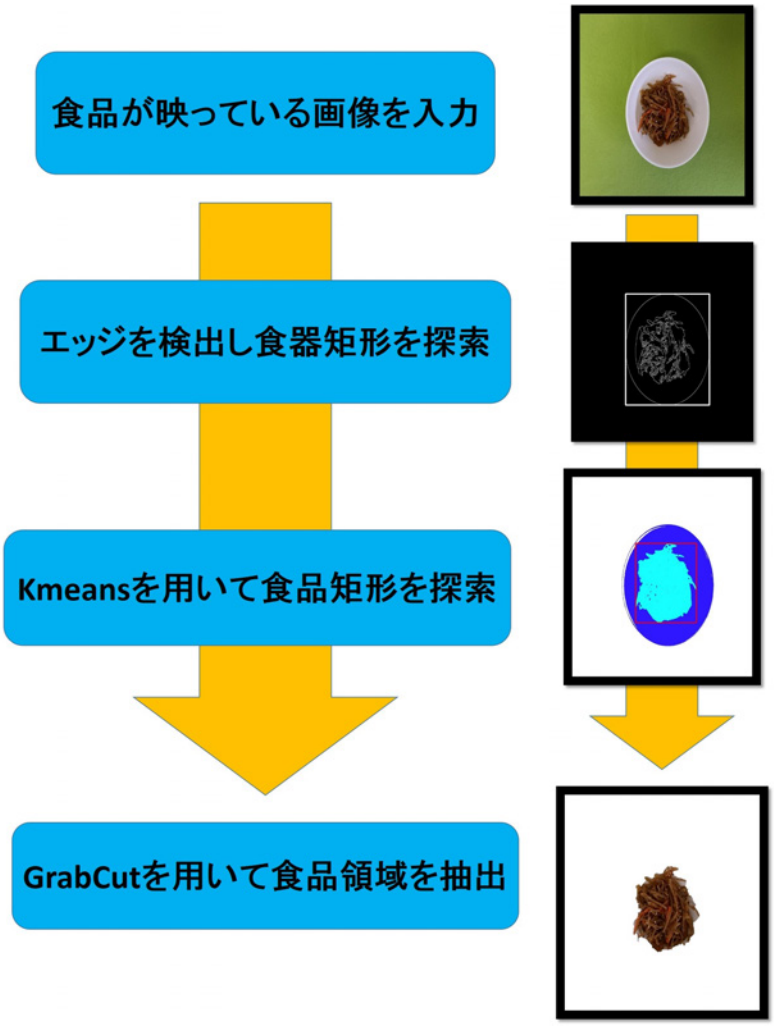

図 4.1: 食品領域抽出のフローチャート

図 4.2: 食器矩形抽出結果

図 4.3: 食器矩形での GrabCut を行う前 (左) と後 (右) の結果

食器上の食品領域を正確に検出できるように, 色情報を頼りに更に分割を行い正確な食品領域抽出を行う.

4.1.2　食品領域抽出

上記までで食器を検出することは行うことはできたが, それだけでは正確なカロリー量などを検出することはできない. そこで, 食器上から更に食品領域の抽出を行う. これには先ほど抽出した食器付近の矩形画像に対して, 画像上の色情報を k-Means[15] を用いてクラスタリングする. この時にはピクセルの RGB 値をデータ点とし, k=3 としてクラスタリングすることで, “食品”, “食器”, “背景” の 3 種類に画像を分け, 正確な食品領域抽出を行う.

しかし単純に k-Means を行うと, 様々な色の具材があるような食品の際に誤った分割を行ってしまう場合がある. そこでこの時, 画像にはメディアンブラーを掛けることで細かな色の差を消し, 食品内の具材などの局所的に色が違うものに関して誤った分割にならないようにしている. また食品は必ず食器の中心付近にあると仮定し, 中心から “食品”, “食器”, “背景” と画像中の領域を定義する. 実際に “食品”, “食器”, “背景” に分割を行い, 食品領域矩形を示した図を以下の図 4.4 に示す.

図 4.4: “食品”, “食器”, “背景” 分割結果 (食品:水色, 食器:青, 背景:白)

4.1.3　GrabCut

上記までで食品領域矩形を求めることができたので, その矩形を GrabCut[16] を用いて分割を行い画像中から正確な食品領域を求める.

GrabCut とはグラフの最小切断によるエネルギーの最小化法であるグラフカットを簡単に扱えるようにしたものである. グラフカットではユーザーがある程度の前面, 背景の指定を行わなくてはならなかったが, GrabCut では領域分割を行いたい部分の矩形を与えるだけでよく, 矩形外は自動的に背景と扱うので指定を行わずとも正確な領域分割を可能にした. 本システムではこの GrabCut を用いて食品及び基準物体領域を抽出する. 実際に求められた食品領域矩形を用いて分割を行った最終的な食品領域抽出結果を図 4.5 に示す.

図 4.5: 食品領域抽出最終結果

4.2　基準物体領域抽出

ここでは基準物体領域抽出について述べる. まず基準物体についてだが, 画像よりサイズ推定を行う場合しばしばチェッカーボードを用い, 対象物体と一緒に撮影, 比較を行う場合が多い. しかし今回は食事を対象とするので出先など様々な場所で撮影されることが予想される. その時にチェッカーボードを常に携帯しなくてはならないのは非常に負担となる. そこで今回は GrabCut を用いることでユーザーが常に携帯しているものを基準物体とすることが出来るシステムを提案する.

4.2.1　抽出方法

基準物体の抽出には食品領域抽出と同じく GrabCut 及びエッジ検出を組み合わせて用いる. この際に単純にエッジ検出のみで基準物体領域抽出の実装も考えたが背景のノイズや, 画像の映り方などでエッジが検出されない場合があった. そこで GrabCut も同時に使用することで精度の向上を図った. また GrabCut を用いることで, 基準物体は事前に面積さえわかればどのような形状のものでも使用することができ, ユーザー各々が常に携帯しているものを使用することができるので非常に利便性が高くなっていると考えられる. 撮影時に真上以外の方向から撮影すると, 射影歪みが生じ, 画像上での基準物体と食品領域の面積比が実際の面積比を正しく反映しなくなるため, 正しい食品カロリー量の推定が困難になってしまう. 今回のシステムでは, この問題に対して補正は行わなわず, 必ずテーブルの真上から食事画像と基準物体を撮影するという前提を置くことで対処する.

実際の基準物体の例としてゲームセンターのカード及び長財布を挙げ, 領域抽出結果を以下の図 4.6, 4.7 に示す.

図 4.6: 基準物体の例 (カード)(左) と基準領域抽出の結果 (右)

図 4.7: 基準物体の例 (長財布)(左) と基準領域抽出の結果 (右)

4.3　カロリー量推定

カロリー量推定では3D 復元を行ったりする場合があるが, これには複数視点からの画像が必要となり, ユーザーの負担になる場合がある. そこで今回はより簡単にユーザーに使用してもらうため 1 枚の画像より食品の面積を求め, そこから食品のカロリー量を推定する.

上記の方法を用いて食品領域及び基準物体領域を抽出したのち, 基準物体の大きさとの比較を行い食品領域の面積を求める. そして, 求められた食品の面積より食品のカロリー量を推定する. 食品の面積はすでに大きさのわかっている基準物体の面積より求める. 食品領域, 基準物体領域それぞれのピクセル数を数え, 食品領域のピクセル数を基準物体領域のピクセル数で割ることで大きさの比率を取得する. そして基準物体の面積とその比率を掛けあわせることで食品領域の面積を求めている. 基準物体のピクセル数を B_p, 面積を B_a, 食品領域のピクセル数を F_p, 面積を F_a, とした時食品の面積は以下の式 4.1 で求めることができる.

$$F_a = B_a * \frac{F_p}{B_p} \tag{4.1}$$

上記の式で食品の面積を求めることができたのでこの面積を用いて食品のカロリー量を推定する. この時, 面積から線形にカロリー量を推定する方式では, 唐揚げや餃子といった個数でカロリー量が変わったり平面的に盛りつけるものに対しては有効であると考えられるが, ご飯や丼料理のような深さのあるものに盛られることが多い食品に対して, 正確な認識ができない. そこでカロリー量推定では事前に複数サイズの食品のカロリー量より学習しておいた回帰曲線に基づき, 推定を行う. 回帰曲線を用いることで, 牛丼やごはんといった深さのあるものに盛られることが多い食品に関しても 2 次曲線のようなフィッティングを行うことができるのでカロリー量を推定することが出来る. つまり食品のカロリー量を C とした時, カロリー量は以下の式 4.2 で求められる. a, b, c の係数についてはそれぞれ食品ごとに事前に学習データセットより学習をしておく.

$$C = a * F_a^2 + b * F_a + c \tag{4.2}$$

4.4　認識エンジン

4.4.1　Deep Convolutional Neural Network

本システムでは, ディープラーニング (DCNN) を画像認識に用いている. これは既存のアプリなどとは異なりサーバとの通信などは行っておらず, すべてアプリ内で完結している [17].

現在最もポピュラーなディープラーニングのネットワークは “AlexNet” [18] であり, 様々な問題において良い結果を示している. しかし, これはパラメータ数が非常に多くメモリを必要とし計算時間もかかってしまうので, 今回のモバイルアプリという形式には適していない.

そこで今回は “Network in Network” (NIN)[19] という形式を用いて, パラメータ数を抑え, 計算時間の短縮を行いモバイルでの実行を可能にしている. AlexNet は 6000 万パラメータが必要なのに対して, NIN は 750 万パラメータで同程度の認識性能を実現している. さらに, パラメータを 4bit に圧縮することで, 3.8MB 程度で学習パラメータを保持できた. これは従来の Fisher Vector と SVM 重みの圧縮を用いた場合 [20] のパラメータサイズ 9MB よりも大幅に小さいサイズとなっており少メモリで非常にモバイルに適した形となっている. 高速化部分ではマルチコア対応, NEON 命令での記述, 画像の縮小などの手法を行うことで, SAMSUNG Galaxy S5(クアッドコア 2.5GHz) において約 0.2 秒程度での実行が可能となり, リアルタイムでの処理も可能な速度とすることができた.

学習には DCNN は 4 つの畳み込み層を持った NIN のモデルを利用した. 学習には Caffe[21] を利用して, 最初にスクラッチから ILSVRC1000 種類と食事関連した 1000 種類の ImageNet 画像の 2000 クラス 210 万枚で学習し, そのモデルを UEC-FOOD100[22] の食事 100 クラスと, 主に Twitter から収集した食事写真と間違えやすい非食事画像 1 万枚の合計 101 クラスでファインチューニングした.

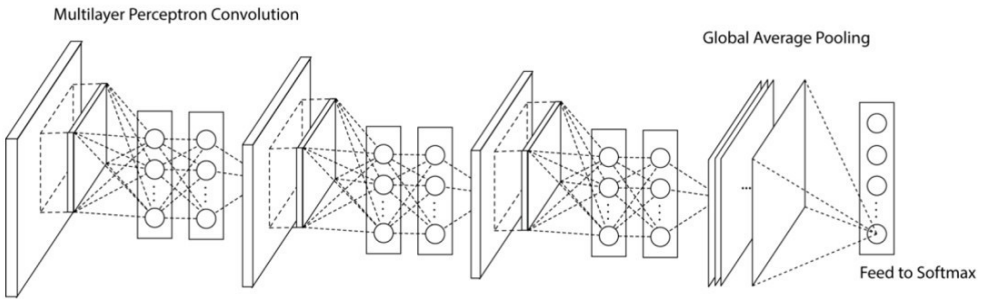

図 4.8: Network in Network の構成図 ([19] より引用)

4.4.2　性能評価

認識エンジン単体での性能評価をここで行う. 認識例を図 4.9 に示す. また非食事を除いた 100 クラス分類の評価を 5fold cross validation で行った結果を図 4.10 に示す. パラメータ圧縮なしの場合, 認識率は 1 位で 75.0%, 5 位以内で正解は 93.7%と高い性能を示していることがわかる.

図 4.9: 認識結果例

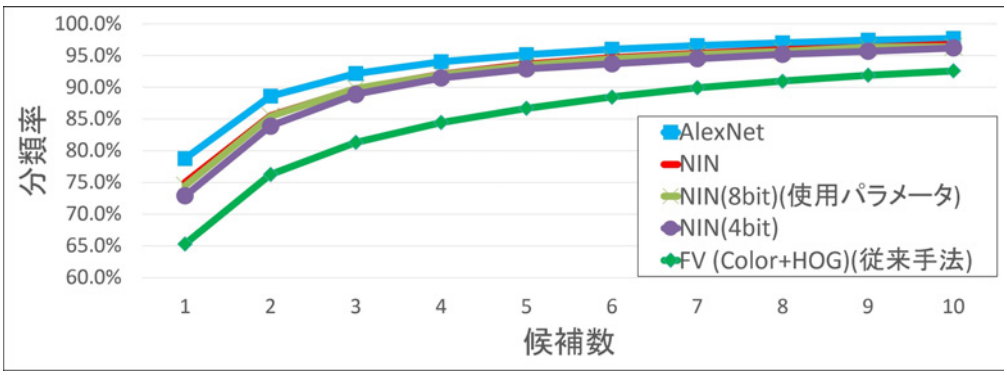

図 4.10: 認識エンジンの性能評価図

第5章

ユーザーインタフェース

ここではユーザーインタフェースの実装について記述する．Android は Java で開発するのが一般的だが, 認識エンジンは高速性が重要なので, C 及び C++を用いて認識部分や領域分割部分を記述した．そしてその部分を Java Naitve Interface(JNI) を使って Java で書かれているアプリ本体から呼び出す形をとった．JNI を用いることで拡張性の高さと高速化を実現した．またモバイルという利点を活かし, 単品の認識のみならず, デバイスを動かしながら撮影することで複数品でのカロリー量推定を可能にした．撮影時には食品や基準物体を真上から撮影することを想定している．またスマートフォン上ではカメラのアスペクト比の都合上横方向に潰れているが, これは食品, 基準物体一律同じ潰れ方をしているので大きさ推定に影響はない．

5.1　単品認識モード

単品認識モードでは画像内に基準物体と食品を映すことで, カロリー量を推定することが出来る．ラーメンや丼料理など, 主に単品で食べるものに関してはこちらのモードを使用することを想定してる．単品認識モードでの使用画面を以下の図 5.1 に示す．

5.2　複数品認識モード

モバイルではその特徴上動かしながら撮影を行うことが出来る．そこでその特性を活かし, 一度基準物体の大きさを取得し, その撮影した高さを維持して他の食品を撮影することで, 複数品のカロリー量を推定することが可能である．これは定

図 5.1: 単品認識モードでの実行例

食などの複数品を同時に食べるようなメニューに適用できると考えている. 以下の図 5.2 では使用風景例を, 図 5.3 及び図 5.4 で基準物体の登録画面を示す. さらに図 5.5, 図 5.6 で実際の複数品認識モードでの実行例を示す. また図 5.2 の風景画像と図 5.3〜図 5.6 のスマートフォン画面をキャプチャしたもので基準物体の大きさなどが異なって見えるがこれは前述の通りスマートフォンの仕様上であり, 一律に食品, 基準物体の画面上の大きさが変わっているので大きさ推定に影響はない.

図 5.2: 複数品認識モードの使用風景例

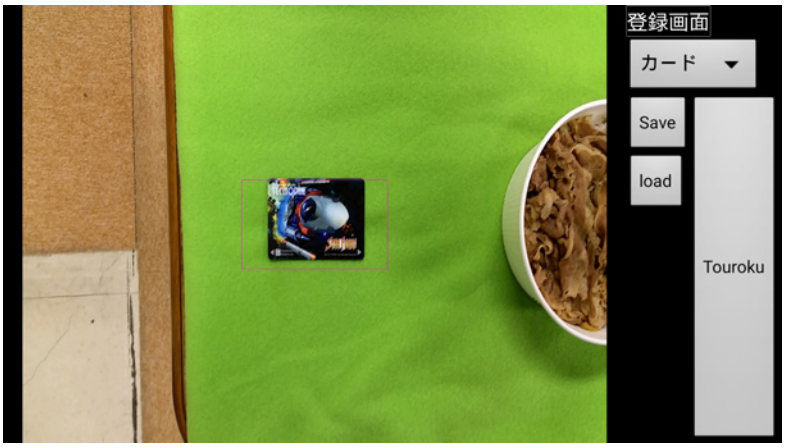

図 5.3: 基準物体の大きさ登録例 1

図 5.4: 基準物体の大きさ登録例 2

図 5.5: 複数品認識モードの実行例 1

図 5.6: 複数品認識モードの実行例 2

第6章

実験

6.1　データセット

今回使用する食品は, スーパーやコンビニエンスストア, オリジン弁当などのお惣菜屋さん, 生協食堂などパッケージや販売会社の HP にカロリー量情報が記載されているものを使用した. 対象食品は日本食を中心とした 20 種類を選択した. 今回使用した食品を以下の表 6.1 に示す.

データセットは 1 食品 3 サイズごと 1 枚ずつのカロリー量のわかっている画像 60 枚を用意した.　今回作成したデータセットの画像の一部及び, 面積, カロリー量は付録 A に記述する. 実験は精度評価用の大量のデータをスマートフォンで処理するのは難しいため, カロリー量推定実験を PC 上で行い, ユーザー評価実験をスマートフォンで行う. 学習用データセットの一例を図 6.1 に示す.

図 6.1: データセットの画像例

表 6.1: 実験に使用した食品

ごはん	ひじきの煮物	サラダ	ナポリタン	お浸し
牛丼	肉じゃが	酢豚	春巻き	コロッケ
唐揚げ	きんぴら	エビチリ	餃子	たこ焼き
焼きそば	トンカツ	焼き鳥	筑前煮	フライドポテト

6.2　カロリー量推定

6.2.1　方法

学習用の画像として 1 食品あたり 3 サイズの基準物体と共に写ったカロリー量付き画像を用意し, その画像より求められた面積とカロリー量より回帰曲線を求める. そしてテスト用画像より求められた面積を回帰曲線に当てはめカロリー量を推定し, 実際のカロリー量との差を計測する. テスト用画像も 1 食品あたり 3 サイズ用意し, 合計 60 枚での実験を行った. テスト用画像に使用した食品は, 学習用画像とは別のスーパーやコンビニ, 外食チェーンで購入したものを撮影し使用している. テストセットでの画像例を以下の図 6.2 に示す. また今回作成したテストセットの画像の一部及び, 面積, カロリー量についても付録 A に記述する.

図 6.2: データセットの画像例

6.2.2　結果

カロリー量推定結果を以下の表 6.2, 及び全体の結果を表 6.3 に示す. 正解カロリー量と推定カロリー量間の絶対値を誤差と定め平均をとったものを誤差平均と

し, 加えて平均二乗誤差平方根 (RMSE) での評価も行った. また誤差がどの程度の割合かを調べるために推定された誤差を実際のカロリー量で割ったものを相対誤差として求める. さらに誤差の標準偏差, 標準偏差を誤差平均で割ったものを相対標準偏差として求める. 表 6.3 ではそれぞれの平均を示している. 表中の単位は相対誤差平均以外は kcal とする. また全食品, 全サイズの推定結果は付録 B に記述する.

表 6.2: カロリー量推定結果

	正解カロリー量			推定カロリー量			誤差平均	RMSE	標準偏差	相対誤差平均	相対標準偏差
食品名	S	M	L	S	M	L					
ごはん	221	340	459	147.56	126.44	438.06	102.65	130.95	81.3	0.335	0.792
ひじき	32.94	51.76	80	31.09	37.91	61.18	11.51	13.53	7.12	0.186	0.619
サラダ	6	10	14	12.38	9.04	13.59	2.58	3.73	3.34	0.396	1.294
ナポリタン	278	441.49	605	322.18	334.7	362.84	131.04	154.92	116.96	0.267	0.893
お浸し	30	54	81	34.39	47.08	79.63	4.23	4.8	4.62	0.097	1.092
牛丼	737	962	1322	665.53	1032.83	1417.54	79.28	80.11	73.6	0.081	0.928
肉じゃが	120	170	276	166.32	122.19	116.19	84.65	99.95	84.26	0.415	0.995
酢豚	195	292	476	148.27	201.24	222.89	130.2	157.57	88.75	0.361	0.682
春巻き	214	428	642	150.51	377.16	552.67	67.89	69.75	16.02	0.185	0.236
コロッケ	184	368	552	212.77	357.45	513.96	25.79	28.2	27.42	0.085	1.063
から揚げ	97.6	292.8	488	101.16	234.88	386.54	54.31	67.48	43.08	0.147	0.793
きんぴら	36	72	108	57.34	69.57	85.04	15.57	18.15	18.1	0.28	1.163
エビチリ	178.42	267.63	368	263.07	291.82	319.56	52.43	58.01	54.41	0.232	1.038
餃子	82	246	494	65.85	202.81	457.94	31.8	33.8	11.44	0.148	0.36
たこ焼き	160.5	240.75	321	159.51	270.23	354.5	21.32	25.77	15.4	0.078	0.722
焼きそば	188.89	283.33	425	241.23	395.45	519.01	86.16	89.72	25.03	0.298	0.29
トンカツ	120.75	345	586.5	110.04	318.8	559.65	21.25	22.52	7.46	0.07	0.351
焼き鳥	55.2	165.6	276	51.21	142.79	179.96	40.95	57.04	39.71	0.186	0.97
筑前煮	56.68	85.02	124	32.28	98.15	137.51	17.01	17.8	17.78	0.231	1.045
フライドポテト	249	454	571	339.47	469.55	484.95	64.02	72.64	72.34	0.183	1.13

表 6.3: 全体の平均

全体誤差平均	RMSE 平均	標準偏差平均	相対誤差平均	相対標準偏差平均
52.231	60.322	40.401	0.213	0.823

今回の実験では 20 種類の食品で約 50kcal 程度の誤差に収めることができた. 相対誤差においても約 20 %程度となっている.

6.3　ユーザー評価

6.3.1　方法

被験者には, 河野らが提案したシステム “FoodCam” [1] と今回提案したシステムでそれぞれ実際にカロリー量を記録してもらい, 実際のカロリー量とどれだけ誤差が出たのかを計測し, 更に使用感の評価を行ってもらう. 実験では 12 人の栄養などの知識があまり無い被験者に使用してもらった. 対象食品は “牛丼”, “コロッケ”, “サラダ” の 3 種類とした. 対象食品を以下の図 6.3, 図 6.4, 図 6.5 に示す.

図 6.3: ユーザー評価で使用した食品 1(牛丼:962kcal)

図 6.4: ユーザー評価で使用した食品 2(コロッケ:552kcal)

図 6.5: ユーザー評価で使用した食品 3(サラダ:14kcal)

6.3.2　使用例

今回提案したシステムではガイドに合わせ基準物体と共に撮影するだけでカロリー量を推定すること出来る．一方 FoodCam に関してはスマートフォンに食品を映しながら, 画面右下のスライダーを用いて食品の大きさを使用者が主観的に評価することでカロリー量を推定することが出来る．この時スライダーでの大きさ評価は 10 段階での評価となっている．以下の図 6.6 に FoodCam のキャプチャ画面を示す．

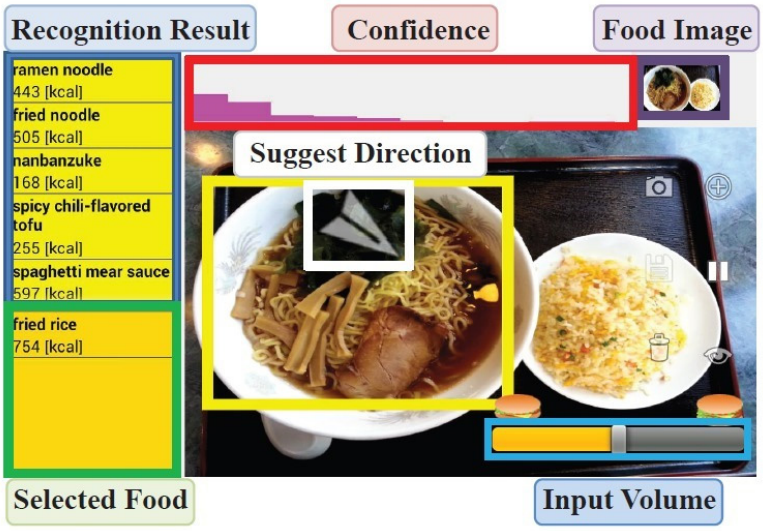

図 6.6: FoodCam のキャプチャ画面 ([1] より引用)

6.3.3　結果

実際に被験者 12 名に 2 種類のシステムを使用してもらい測定されたカロリー量誤差の平均及び標準偏差を以下の表 6.4, 表 6.5 に示す．表中の単位は全て kcal とする．またユーザーに実際に推定してもらったカロリー量の全データは付録 B に記述する．

表 6.4: カロリー量をユーザーが実際に測定した結果 (FoodCam)

		FoodCam		
食品名	実際のカロリー量	推定カロリー量平均	実際との差	標準偏差
牛丼	962	908.75	-53.25	209.79
コロッケ	552	310	-242	91.26
サラダ	14	68.83	54.83	36.28

表 6.5: カロリー量をユーザーが実際に測定した結果 (提案システム)

		提案システム		
食品名	実際のカロリー量	推定カロリー量平均	実際との差	標準偏差
牛丼	962	720	-242	55.1
コロッケ	552	504.92	-47.08	52.52
サラダ	14	18.86	4.86	11.87

結果より牛丼では FoodCam よりカロリー量誤差が大きくなってしまったが, コロッケ, サラダにおいては FoodCam より大幅にカロリー量誤差を小さくすることができた．また標準偏差においては全ての食品において FoodCam を下回る値となっており, 栄養などの知識の無いユーザーでもブレの少ないシステムとすることができた.

またカロリー量の記録の取りやすさについて 1 を取りにくい, 3 を普通, 5 を取りやすいとした 5 段階評価を行ってもらった結果の平均及び標準偏差を以下の表 6.6 に示す.

表 6.6: システムのカロリー量記録の取りやすさ

	記録の取りやすさ
FoodCam[1]	2.83±0.80
提案システム	4.25±0.72

結果として既存の FoodCam よりも記録の取りやすいという評価を得た. FoodCam では食品をスマートフォンに映しながらスライダーを変化させなければならず, 慣れていないユーザーにとっては少し難しいシステムとなっていた. 今回の提案システムでは, 真上より基準物体と共に撮影を行うだけなので, 非常に操作も単純だったことが記録の取りやすさにつながったのだと考えられる.

第7章

考察

7.1　カロリー量推定の考察

今回は全体の精度で, カロリー量誤差平均が 52.231kcal, 相対誤差が 0.213 という値になった. この 0.213 という値は消費者庁の定める食品に表示するカロリー量の誤差範囲 20 % [1] に極めて近い値なので, 非常に有用性が高いものと考えられる.

個別の精度ではたこ焼きや, コロッケ, トンカツといった個数によってカロリー量が変化するような食品に関しては非常に高い精度を示していることがわかる. これは面積と個数の間には強い正の相関があり, その影響で面積からおおよその個数を想定することができ精度が高くなったものだと考えられる. 逆に精度が低かったのは肉じゃがや酢豚といった食品であった. これはお椀のようなものに盛られており, 面積だけでは推定が難しかったことと, 作成する企業によって食品中の具材の偏りがあり, 企業ごとのカロリー量の変化が大きかったためであると考えられる. 今後はより様々なデータを取得し, より精度を高めたい. 認識結果の良かったコロッケの画像例を図 7.1, 認識結果の悪かった肉じゃがの画像例を図 7.2 に示す.

図 7.1: 実験に使用したコロッケ画像郡

[1] http://www.caa.go.jp/foods/index4.html

図 7.2: 実験に使用した肉じゃが画像郡

今回はデータセットが非常に少ないにも関わらず, 誤差が 50kcal 程度と良い精度を示すことができた. 今後データセットを拡張していけばより良い精度を得られると考えられる. しかしデータセットの拡張は非常に難しい問題である. 単純な食品画像は大量にインターネット上に存在しているが, カロリーデータ付き画像は非常に少なく, インターネット上から大量の画像を集めることが困難である. Google がカロリーデータ付き画像データセットを公開するとしているが [14], 実際には 2016 年 1 月現在公開はされていない [2]. 今後どのようにデータセットを拡張, 改善していくかが大きな問題であると考えている.

7.2　ユーザー評価の考察

ユーザーに実際にシステムを使ってもらい, コロッケ, サラダにおいては既存システム “FoodCam” よりカロリー量推定誤差を大きく減らすことができた. 牛丼に関しては, FoodCam よりも誤差が大きくなってしまったが, これは FoodCam において一番初めに表示されるカロリー量が 909kcal であり, それを多くのユーザーが選択したため, 正解カロリー量の 962kcal と近い値になったので, 提案システムのカロリー量誤差よりも小さくなったものだと考えられる. 栄養に知識のないユーザーは最初に提示された情報を基にカロリー量を推定することが多いのでこのような結果になったと考えられる. 実際にサラダにおいても正解カロリー量は 14kcal であるにも関わらず, FoodCam で始めに表示されるカロリー量が 131kcal と非常に高い. しかしユーザーはそれを基準として大体こんなものなのだろうとカロリー量を決めてしまい, カロリー量を多く見積もってしまうユーザーが多かった. 提案システムでは面積に応じたカロリー量を事前に学習しているので, ユーザーの知識の有無や主観的評価に影響されることがなくカロリー量誤差を抑えることができた.

[2] https://storage.googleapis.com/food201/food201.zip

また標準分散を見ても FoodCam ではユーザーの主観が入ってしまうので牛丼では 200kcal 以上, 全体を見てもかなり大きいブレが出てきてしまっている. しかし提案システムでは 1000kcal 近いものを推定しているにも関わらず, 50kcal 程度のブレに抑えることができており, 全体としても少ないブレとなっている. これはユーザーの知識や主観評価に関わらず一定の基準で推定ができていることを示している.

カロリー量記録の取りやすさに関しても, FoodCam よりも簡単な操作でカロリー量を推定することができたので, 提案システムの方が記録を取りやすいという評価を得た. 今後改善点としては誤認識が起きた際には多少手動で認識結果を修正し, より良い精度になるようなシステムを作成すればより使用感の評価も高くなると考えられる.

提案システムの推定精度及び記録のとりやすさが既存システムよりも有意に優れていることを調べるために t 検定を行い結果を以下の表 7.1 にまとめた. 有意水準は 5 %で検定を行った. また t 境界値の両側の値は 2.200 となっている. つまり t 値の絶対値が 2.200 よりも大きければ差が有意であると言える.

表 7.1: 食品ごとの t 検定結果

	牛丼	コロッケ	サラダ	記録の取りやすさ
t 値	3.012	-6.066	4.438	-4.214
p 値	0.011	$8.118e^{-5}$	$9.981e^{-4}$	$1.450e^{-4}$

表 7.1 より全ての食品において差が有意であることがわかった. 牛丼ではカロリー量の誤差が大きくなかったため少し差が小さくなっているが, コロッケやサラダに関しては非常に差が大きくなった. また記録のとりやすさにおいても有意差が認められたので, 既存システムよりもカロリー量を記録する部分では非常に有用なシステムの作成ができたことがわかる.

第8章

終わりに

8.1　まとめ

本論文では, 食品と基準物体を同時に映すことで画像中より自動的にカロリー量推定を行うシステムを作成した. 食品領域抽出には GrabCut を用いた. しかし単純に GrabCut を画像に適用すると背景を多く含んでしまい全く使えないものであった. そこで食器の検出にエッジ検出を, 食品領域抽出に k-Means を用いることで正確な食事領域抽出を実現した. 画像認識には DCNN を用いた. 計算量が非常にかかる部分であったが C 言語での記述や NIN の採用, マルチスレッド化などにより高い精度を維持したまま約 0.2 秒程度の実行速度を実現した. 実験ではカロリー量推定実験とユーザー評価実験の 2 つを行った. カロリー量推定実験での誤差の平均は 52.231kcal, 相対誤差の平均は 0.213 となり, ユーザー評価実験でも既存システムよりも記録を取りやすいという評価を得た.

8.2　今後の課題

現在ではほぼ柄のないランチョンマット上での実行を行っており, そのためエッジ検出での食器検出がうまくいっているが, 実際の環境下ではテーブルの木目や模様, 広告などでエッジがうまく検出されず正しい食器検出ができない場合がある. 今後は楕円フィッテングなどを用いてより背景情報に左右されない食器検出手法を考える必要がある.

また, 食事領域抽出においても色情報を主に使用しているので, 食品と食器の色が類似している場合やカレーライスのように大きく色の異なる部分がある食品に

はうまく食事領域を抽出できない場合が存在する (図 8.1). 今後はテクスチャ特徴などを用いてより正確に食事領域を抽出したい.

図 8.1: カレーライスでの食品領域分割実行例

今後は複数品においてもまとめて撮影し, 一回の撮影で食事全体のカロリー量がわかればより有用性が上がると考えられる (図 8.2). その際にはユーザーに大まかな食品領域を指定してもらい, その情報を用いて GrabCut を行うなどをすれば, ユーザーの負担を最小限に抑えつつ, 実行が可能であると考えられる. 現在の方式にとらわれず今後も様々な手法を試していきたい.

今回は画像中より面積を求め, それを一定の単位の下, 積算よりカロリー量を求めたが実際には面積ではなく, 体積を求めたほうがより正確なカロリー量を推定できると考えられる. 体積の推定には, 複数枚の異なる視点での画像を撮影し, 3D 復元を行う方法が考えられるが, あまり操作が多くなると気軽に使用できずに使ってもらえなくなる場合もあるので使用感と, 精度の良いバランスを考え選択しなければならない.

図 8.2: 想定される実際の使用環境

参考文献

[1] Y. Kawano and K. Yanai. Foodcam-256: A large-scale real-time mobile food recognition system employing high-dimensional features and compression of classifier weights. In *Proceedings of the ACM International Conference on Multimedia*, pp. 761–762, 2014.

[2] K. Okamoto and K. Yanai. Grillcam: A real-time eating action recognition system. In *Proc. of International Conference on Multimedia Modelling (MMM)*, 2016.

[3] 株式会社クオリア. http://www.qualia.tokyo.

[4] K. Aizawa, Y. Maruyama, H. Li, and C. Morikawa. Food balance estimation by using personal dietary tendencies in a multimedia food log. *IEEE Transactions on Multimedia*, Vol. 15, No. 8, pp. 2176–2185, 2013.

[5] W. Wu and J. Yang. Fast food recognition from videos of eating for calorie estimation. In *Proc. of IEEE International Conference on Multimedia and Expo(ICME)*, pp. 1210–1213, 2009.

[6] 特定非営利活動法人日本栄養改善学会. 食事調査マニュアル 改定 2 版. 南山堂, 2008.

[7] W. Shimoda and K. Yanai. Cnn-based food image segmentation. In *Proc. of International Workshop on Multimedia Assisted Dietary Management (MADIMA)*, 2015.

[8] J. Dehais, M. Anthimopoulos, and S. Mougiakakou. Dish detection and segmentation for dietary assessment on smartphones. In *Proc. of International Workshop on Multimedia Assisted Dietary Management (MADIMA)*, pp. 432–440, 2015.

[9] T. Miyazaki, Gamhewage C. De S., and K. Aizawa. Image-based calorie content estimation for dietary assessment. In *IEEE International Symposium on Multimedia*, pp. 363–368, 2011.

[10] P. Pouladzadeh, S. Shirmohammadi, and R. Almaghrabi. Measuring calorie and nutrition from food image. *IEEE Transactions on Instrumentation and Measurement*, pp. 1947–1956, 2014.

[11] F. Kong and J. Tan. Dietcam: Automatic dietary assessment with mobile camera phones. In *Proc. of Pervasive and Mobile Computing*, pp. 147–163, 2012.

[12] C. Xu, Y. He, N. Khannan, A. Parra, C. Boushey, and E. Delp. Image-based food volume estimation. In *Proceedings of the international workshop on Multimedia for cooking & eating activities*, pp. 75–80, 2013.

[13] M. Chen, Y. Yang, C. Ho, S. Wang, S. Liu, E. Chang, C. Yeh, and M. Ouhyoung. Automatic chinese food identification and quantity estimation. In *SIGGRAPH Asia Technical Briefs*, p. 29, 2012.

[14] A. Meyers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. P. Murphy. Im2calories: Towards an automated mobile vision food diary. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[15] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, Vol. 28, pp. 100–108, 1979.

[16] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts. In *Proc. of ACM SIGGRAPH*, pp. 309–314, 2004.

[17] 岡元晃一, 柳井啓司. Deepfoodcam: Dcnn による 101 種類食事認識アプリ. 画像の認識・理解シンポジウム (MIRU), 2015.

[18] A. Krizhevsky, S. Ilya, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105. 2012.

[19] M. Lin, Q. Chen, and S. Yan. Network in network. *In Proc. of International Conference on Learning Represenation Conference Track*, 2013.

[20] Y. Kawano and K. Yanai. ILSVRC on a Smartphone. *IPSJ Transactions on Computer Vision and Applications*, Vol. 6, pp. 83–87, 2014.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proc.of ACM International Conference Multimedia*, pp. 675–678, 2014.

[22] UEC FOOD 100. http://foodcam.mobi/dataset.html.

付 録A

今回作成したデータセット

今回収集したデータセット及びテストセットの画像の一部を以下の図に示し, 全体の面積及びカロリー量を以下の表に示す. 画像は全て左側より S,M,L サイズの順番で並んでいる.

図 A.1: ごはん　データセット画像

図 A.2: ごはん　テストセット画像

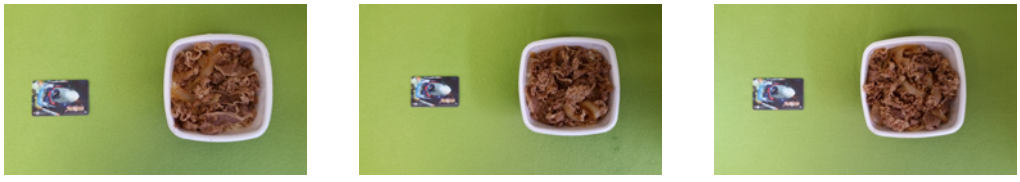

図 A.3: 牛丼　データセット画像

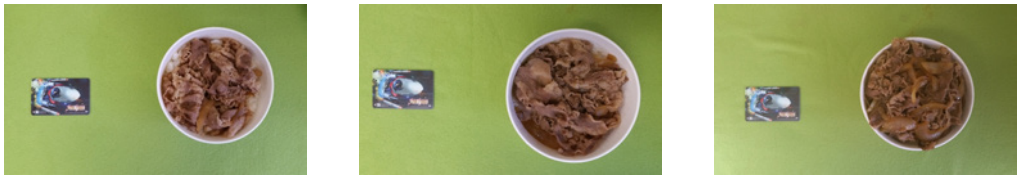

図 A.4: 牛丼　テストセット画像

表 A.1: データセットの面積及びカロリー量

	データセット面積 (cm^2)			データセットカロリー量 (kcal)		
食品名	S	M	L	S	M	L
ごはん	126.03	178.09	195.85	142	242	285
ひじき	66	98.7	129.89	44	73	100
サラダ	103.68	171.02	184.78	8	13	17
ナポリタン	289.8	501.01	597.04	399	614	748
お浸し	79.27	121.5	132.53	53	80	116
牛丼	184.82	221.54	228.73	669	929	1063
肉じゃが	85.07	116.09	132.9	116	174	275.5
酢豚	70.05	92.31	112.75	150.4	225.6	357.2
春巻き	52.11	89.47	131.83	166	332	498
コロッケ	59.87	111.45	163.81	198	369	594
から揚げ	25.82	75.11	116.93	77.7	259	414.4
きんぴら	63.91	78.08	100.44	53.2	79.8	119.7
エビチリ	58.32	69.84	108.8	141.6	212.4	336.3
餃子	18.38	61.46	96.96	50	150	250
たこ焼き	82.52	114.87	153.81	235.5	353.25	471
焼きそば	133.35	182.67	264.51	232.98	360	678
トンカツ	31.29	80.76	146.47	124	372	744
焼き鳥	31.85	120.92	229.59	63.3	190	380
筑前煮	76.44	81.88	98.84	74.4	94	141
フライドポテト	87.47	136.95	204.4	207	310.5	414

表 A.2: テストセットの面積及びカロリー量

	テストセット面積 (cm^2)			テストセットカロリー量 (kcal)		
食品名	S	M	L	S	M	L
ごはん	129.28	119.69	306.14	221	340	459
ひじき	51.6	59.12	85.05	32.94	51.76	80
サラダ	79.18	97.62	74.23	6	10	14
ナポリタン	163.79	190.37	240.37	278	441.49	605
お浸し	53.78	73.16	110.14	30	54	81
牛丼	192.3	227.16	243.23	737	962	1322
肉じゃが	114.4	99.63	95.01	120	170	276
酢豚	68.79	87.01	91.76	195	292	476
春巻き	48.82	100.44	147.36	214	428	642
コロッケ	65.07	108.52	146.72	184	368	552
から揚げ	26.43	68.21	109.93	97.6	292.8	488
きんぴら	40.5	64.04	92.66	36	72	108
エビチリ	80.31	87.9	97.99	178.42	267.63	368
餃子	25.94	80.91	157.72	82	246	494
たこ焼き	63.86	91.48	115.09	160.5	240.75	321
焼きそば	137.15	193.71	227.72	188.89	283.33	425
トンカツ	28.33	70.63	115.06	120.75	345	586.5
焼き鳥	22.27	90.09	114.9	55.2	165.6	276
筑前煮	66.41	83.17	97.36	56.68	85.02	124
フライドポテト	153.65	267.78	305.12	249	454	571

付 録B

全カロリー量推定結果及び ユーザーによるカロリー量推定結果

今回のカロリー量推定実験の食品ごとの 3 サイズごとのカロリー量誤差平均及び相対誤差を以下の表 B.1 に示す.

またユーザー評価実験での実際にユーザーに使用してもらい, 求められたカロリー量とその誤差を以下の表に示す. 表 B.2 に既存システムの結果を, 表 B.3 に既存システムの結果を示す.

付 録 B　全カロリー量推定結果及びユーザーによるカロリー量推定結果44

表 B.1: 計測されたカロリー量誤差及び相対誤差

	カロリー量誤差 (kcal)				相対誤差			
	S	M	L	平均	S	M	L	平均
ごはん	73.443	213.564	20.943	102.65	0.332	0.628	0.046	0.335
ひじき	1.853	13.845	18.822	11.507	0.056	0.267	0.235	0.186
サラダ	-6.384	0.96	0.405	2.583	-1.064	0.096	0.029	0.396
ナポリタン	-44.177	106.793	242.159	131.043	-0.159	0.242	0.4	0.267
お浸し	-4.394	6.917	1.366	4.226	-0.146	0.128	0.017	0.097
牛丼	71.471	-70.83	-95.537	79.279	0.097	-0.074	-0.072	0.081
肉じゃが	-46.317	47.813	159.814	84.648	-0.386	0.281	0.579	0.415
酢豚	46.728	90.761	253.108	130.199	0.24	0.311	0.532	0.361
春巻き	63.495	50.841	89.33	67.889	0.297	0.119	0.139	0.185
コロッケ	-28.774	10.545	38.037	25.785	-0.156	0.029	0.069	0.085
から揚げ	-3.56	57.917	101.457	54.311	-0.036	0.198	0.208	0.147
きんぴら	-21.336	2.432	22.957	15.575	-0.593	0.034	0.213	0.28
エビチリ	-84.649	-24.188	48.444	52.427	-0.474	-0.09	0.132	0.232
餃子	16.147	43.189	36.057	31.798	0.197	0.176	0.073	0.148
たこ焼き	0.992	-29.48	-33.497	21.323	0.006	-0.122	-0.104	0.078
焼きそば	-52.343	-112.125	-94.013	86.16	-0.277	-0.396	-0.221	0.298
トンカツ	10.71	26.199	26.851	21.254	0.089	0.076	0.046	0.07
焼き鳥	3.991	22.812	96.035	40.946	0.072	0.138	0.348	0.186
筑前煮	24.398	-13.127	-13.509	17.012	0.43	-0.154	-0.109	0.231
フライドポテト	-90.466	-15.545	86.048	64.02	-0.363	-0.034	0.151	0.183

表 B.2: ユーザー評価実験 (FoodCam)

FoodCam						
	求められたカロリー量 (kcal)			カロリー量誤差 (kcal)		
被験者	牛丼	コロッケ	サラダ	牛丼	コロッケ	サラダ
正解カロリー量	962	552	14	0	0	0
A	1272	434	26	310	-118	12
B	727	434	131	-235	-118	117
C	909	248	78	-53	-304	64
D	1090	310	104	128	-242	90
E	909	310	52	-53	-242	38
F	909	310	78	-53	-242	64
G	909	310	52	-53	-242	38
H	727	310	131	-235	-242	117
I	545	186	40	-417	-366	26
J	727	434	30	-235	-118	16
K	909	310	78	-53	-242	64
L	1272	124	26	310	-428	12
平均	908.75	310	68.83	177.91	242	54.83
標準偏差				209.79	91.26	36.28

表 B.3: ユーザー評価実験 (提案システム)

提案システム						
	求められたカロリー量 (kcal)			カロリー量誤差 (kcal)		
被験者	牛丼	コロッケ	サラダ	牛丼	コロッケ	サラダ
正解カロリー量	962	552	14	0	0	0
A	798.5	446.32	7.47	-163.5	-105.68	-6.53
B	718.16	514.43	8.84	-243.84	-37.57	-5.16
C	840.77	561.32	12.12	-121.23	9.32	-1.88
D	693	518.32	31.8	-269	-33.68	17.8
E	722.96	424.94	17.13	-239.04	-127.06	3.13
F	782.6	535.67	12.69	-179.4	-16.33	-1.31
G	665	625.37	30.31	-297	73.37	16.31
H	703.97	450.52	34.02	-258.03	-101.48	20.02
I	685.01	474.96	8.74	-276.99	-77.04	-5.26
J	700.81	520.54	12.83	-261.19	-31.46	-1.17
K	664	494.25	7.44	-298	-57.75	-6.56
L	665.13	492.5	42.88	-296.87	-59.5	28.88
平均	719.9925	504.93	18.86	242.01	60.85	9.50
標準偏差				55.10	52.52	11.87