社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

スマートフォンによる食事画像からの自動カロリー量推定システム

岡元　晃一†　　柳井　啓司†

† 電気通信大学 大学院情報理工学研究科 総合情報学専攻
E-mail: †okamoto-k@mm.inf.uec.ac.jp, ††yanai@cs.uec.ac.jp

あらまし　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.

キーワード　カロリー推定, DCNN 特徴, 画像認識, 食事認識

An automatic calorie estimation of food images on a smartphone

Koichi OKAMOTO† and Keiji YANAI†

† Department of Informatics, The University of Electro-Communications, Tokyo
E-mail: †okamoto-k@mm.inf.uec.ac.jp, ††yanai@cs.uec.ac.jp

1. はじめに

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. そこで本研究では手軽に食事記録を付けることが出来るようなシステムを作成する. これには, 基準物体と食品を同時に撮影するだけでよく, 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ことができるシステムを目指す.

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. システムの使用風景及びキャプチャ画面を図 1 に示す.

図 1　提案システムの使用風景 (左) 及びキャプチャ画面 (右)

2. 関連研究

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 [1] 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. 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 [2] 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 [3] 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 [4] 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 [5] 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 [6] 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 [7] 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.

3. 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. システムのフローチャートを以下の図 2 に示す.

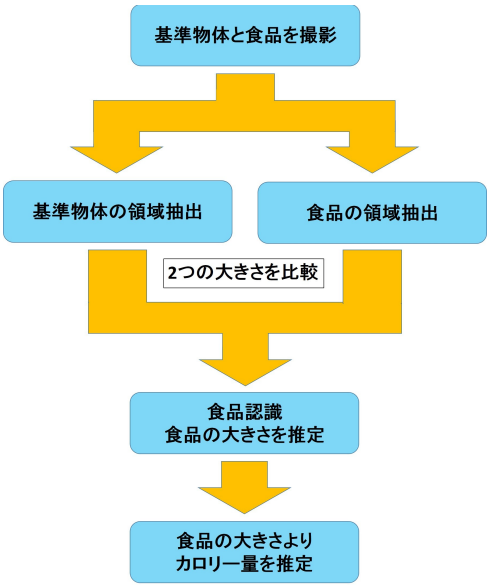

図 2　システムのフローチャート

4. システムの詳細

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 3 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

食品領域を抽出しようと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

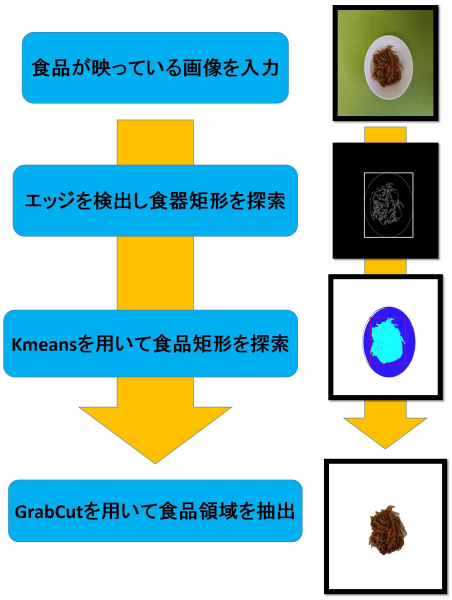

図 3　食品領域抽出のフローチャート

て食器矩形を求めた例が図 4, 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 5 に示す.

図 4　食器矩形抽出結果

図 5　食器矩形での GrabCut を行う前 (左) と後 (右) の結果

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 [8] 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 6 に示す.

図 6　“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 [9] 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 7 に示す.

図 7　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.

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長財布を挙げ, 領域抽出結果を以下の図 8 に示す.

図 8　基準物体の例 (長財布)(左) と基準領域抽出の結果 (右)

4.3　認識エンジン

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 [10].

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” [11] 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) [12] 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 [13] 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った.

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 [14] 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 [15] 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.

4.4　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 1 で求めることができる.

$$F_a = B_a * \frac{F_p}{B_p} \tag{1}$$

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 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.

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

5.　実　　験

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.

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. 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.

5.1　カロリー量推定実験

テストセットも学習用の画像データセットと同様に 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. カロリー量推定の全体の結果を以下の表 1 に示す. 表では全食品の平均値を示している. 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. 表中の単位は相対誤差平均以外は kcal とする.

表 1　全体の平均

全体誤差平均	標準偏差平均	相対誤差平均	相対標準偏差平均
52.231	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.

5.2　ユーザー評価実験

被験者には, 河野らが提案したシステム “FoodCam” [16] 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. 実際に 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 2 に示す. 表中の単位は全て kcal とする.

表 2　ユーザーによる測定結果 (FoodCam)

		FoodCam		提案システム	
食品名	真値	誤差平均	標準偏差	誤差平均	標準偏差
牛丼	962	-53.25	209.79	-242	55.10
コロッケ	552	-242	91.26	-47.08	52.52
サラダ	14	54.83	36.28	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, 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 3 に示す.

表 3　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	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. 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と, 非常に操作が単純だったことが高評価につながったのだと考えられる.

6.　考　　察

6.1　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 %(注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, 作成する食品販売チェーンによって食品中の具材の偏りがあり, 食品販売チェーン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及び悪かった肉じゃがの画像例を図 9 に示す.

図 9　実験に使用したコロッケ画像及び肉じゃが画像

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 [7], 実際には 2016 年 1 月現在公開はされていない(注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.

6.2　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 で初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

(注1)：http://www.caa.go.jp/foods/index4.html
(注2)：https://storage.googleapis.com/food201/food201.zip

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.

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.

7. まとめと今後の課題

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.

今後の課題として, 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. 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. 双方の問題ともに今後はディープラーニングを用いた高精度な検出を行いたい.

現在は真上から撮影のみに対応している. しかし実際には様々な角度の画像でも使用できる方が好ましい. よって真上以外から撮影を行っても基準物体の形状を基に画像を幾何変換し, システムを実行できるようにしたい.

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 10). 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, 現在の方法でも実行が可能であると考えられる. しかし現在ではディープラーニングを用いた高精度な領域分割も研究されている [17]. よって今後は計算量の問題を解決し, 現在の手法にとらわれずこのような手法を取り入れることも考えたい.

作成したシステムは以下のサイトで公開している http://foodcam.jp/calorie/.

図 10　想定される実際の使用環境

文　献

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

[2] 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.

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

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

[5] 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.

[6] 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.

[7] 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.

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

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

[10] 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.

[11] 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.

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

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

[14] 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.

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

[16] Y. Kawano and K. Yanai. FoodCam: A real-time food recognition system on a smartphone. *Multimedia Tools and Applications*, Vol. 74, No. 14, pp. 5263–5287, 2015.

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