# 単一食事画像からの皿と食事の同時分離形状復元

成冨 志優[1,a)]　柳井 啓司[1,b)]

## 概要

近年人々の健康管理意識は高まっており, スマートフォンなどで食事のカロリー量などを管理するアプリケーションが複数存在する. しかしそのようなアプリケーションでは, 雇われている栄養士が手作業でカロリー量や栄養価をつけていたりするため, 時間も人件費もかかる. また画像認識を用いて食事を認識するものも存在するが, 量の測定に手間がかかる問題がある. またそうした既存の画像認識を用いた多くのアプリケーションでは, 面積を推定/測定することでカロリー量を推定しているものの, より正確にカロリー量を求めるには体積が望ましい. そこで本研究では体積の取得を可能にするための手法として, 一枚の食事画像から食事と皿の二つの三次元形状を復元を行うネットワーク, Hungry Networks を作成した. 実現のため食事の三次元モデルのデータセットを作成し, テクスチャ付きの 3D モデルをレンダリングすることで様々な角度から撮影した画像と 3D モデルの対応関係をつくり, CNN を学習することで, 単一画像からの食事と皿の二つの三次元形状を平均 IoU 0.7 程度で高精度に復元することに成功した.

## 1. はじめに

食事の正確なカロリー量推定には食事の量を考慮する必要がある. 現状, 画像認識によってカロリー量を推定するものは, 食事カテゴリや面積を基準に推定するものが多い. しかしより正確なカロリー量推定には面積ではなく体積が推定できることが望ましい. 体積を計算するためには 3 次元形状を測定/推定する必要がある. 近年では深度カメラはスマートフォンに内蔵された機種も登場し, 比較的安価に購入可能になり, 3 次元情報は取得しやすくなった. しかし体積を推定するには深度画像だけでは不足である. なぜなら 1 視点からの深度画像では, 物体の表面の形状はわかっても背面や一部側面は深度カメラに映らないため, 物体全体の三次元形状はわからないからである. そこで深度カメラを用いて体積を得るためには, 複数の視点から撮影し, 合成するの手法が必要になってくるが, カロリー量を知るために食事を様々な角度から撮影するのはユーザにとって手間がかかる. さらに複数視点から撮影することで 3 次元形状を復元しても, 食事以外の物体 (例えば床や近くにある物体など) を切り分ける処理も必要になってくるため後処理が必要になる問題がある. そこで単一の画像から食事の三次元形状復元が実現できれば, カロリー量推定のための体積計算が容易となる. 深層学習を用いて, 画像から直接体積を推定する手法も存在するが, 近年の AR 技術の進歩により, 安価な機器でも画像上の物体の実寸を計算出来る. そのような現状を考慮すると, 深層学習で 3 次元形状を推定し, 単位空間あたりの体積を深層学習で推論して組み合わせることで, 精度よく体積を推論できるようになると考えられる. またカロリー量を正しく推定するためには, 単に食事を三次元復元すればいいわけではない. なぜならカロリー量推定のためには, 食器を取り除いた食品部分のみの体積が重要であるからである. そこで本研究では食事の写った単一の RGB 画像から食事と食器の 2 つの 3 次元形状を復元する Hungry Networks を提案し, 実験によって単一食事画像からの 3 次元形状復元が高精度に可能であることを示す.

[1] 電気通信大学院 情報理工学研究科 情報学専攻
[a)] naritomi-s@mm.inf.uec.ac.jp
[b)] yanai@cs.uec.ac.jp

## 2. 関連研究

画像から 3D 形状を再構成する手法は大きく分類すると 3 つある. ボクセル, 点群, Mesh のどの 3 次元表現で再構成するかである. ボクセルを出力する手法 [2], [14], [16] は GPU のメモリを非常に使うため, 低い解像度でしか再構成ができない. ボクセル表現で高い解像度の出力を得ようとする場合, 実装が非常に複雑になる. 点群の出力表現 [3] はただ単に点の集合を出力するため, プログラム上で再構成した物体の形状を得るためには点同士の接続を別途計算しないといけない. Mesh 表現での出力は主に Mesh テンプレートを使用する手法 [6], [13], [15] が多いが, Mesh テンプレートを必要としない Occupancy Networks [11] や動的に Mesh テンプレートを出力する Mesh R-CNN [5] などがある. Mesh 表現は点とそれらの接続エッジと面から構成されるので, ボクセルに比べメモリ効率よく高解像度にでき, 点群と違い点同士の接続情報もあるので形状も取れるなど, 利点が多い.

## 3. 手法

最終的には食事の体積を求めたい. そのためには体積をもっとも求めやすい三次元表現で三次元形状を復元することが望ましい. 3 次元形状の復元には主にボクセル, 点群, Mesh の 3 つの手法があるが, Mesh 表現が今回の目標にはもっとも適している. ボクセルで体積を求めるためには 1 ボクセルあたりの体積がわかればよいが, 高解像度でなければ正確ではない上, リスケールに弱い. また点群は点と点の接続がわからないため, 体積を推定するのは困難

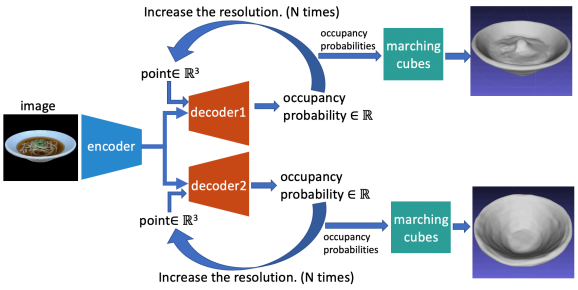


図 1　Hungry Networks の手法概要図

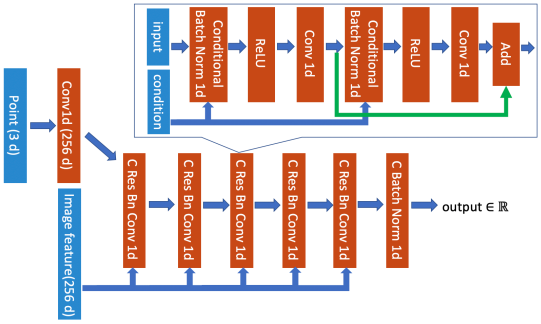


図 2　Hungry Networks のデコーダの構成

である．しかし Mesh は点と点の接続がわかるため,3 次元形状が把握しやすい．Mesh が水密であり自己交差をしていない場合, 頂点が $v \in \mathbb{R}^3$, 各面 $f \in Faces$ が面の表から見た時に反時計回りに $(v1, v2, v3)$ から構成される時, 以下の式で体積を求めることが出来る．

$$\sum_{f \in Faces} \det \begin{vmatrix} v1 & v2 & v3 \end{vmatrix} \qquad (1)$$

Occupancy Networks [11] はオブジェクトの境界面をより高い解像度にしていくアルゴリズムであるため, 殆どの場合で水密かつ自己交差の無い Mesh を単一画像から生成できる．そのため今回の最終的な目標に非常に合致しているので, この手法をベースに, Hungry Networks を作成した．Occupancy Networks では単一の画像から 1 つの Mesh を出力していたが、Hungry Networks では単一の画像から 2 つの Mesh を出力するよう設計した。

### 3.1　Hungry Networks の構成

提案手法である Hungry Networks の概要図を図 1 に示した．個々のエンコーダやデコーダ OSS として公開されている Occupancy Networks の中のものの一部を利用した．入力画像のエンコーダとして ResNet18 を用い, 最終出力層を Global Average Pooling [9] を用いて 1 次元のベクトルにする．エンコードされた特徴量と, 3 次元の座標をデコーダに入力し, 入力した座標の占有確率を求める．デコーダのネットワークを図 2 に示した．図 1 の decoder1 では食事の生成を, decoder2 では食器の生成をするための占有確率をそれぞれ学習する．Mesh 生成時には,Occupancy Networks と同様に, $32 \times 32 \times 32$ の初期解像度で占有確率を求め, 生成したい物体の境界部分のみの解像度を高め, 高い解像度で物体の境界部分のみの占有確率を再度求める．解像度を一度上げるごとにグリッドを 8 分割し, $32 \times 32 \times 32 \Rightarrow 64 \times 64 \times 64 \Rightarrow 128 \times 128 \times 128$ のように解像度を上げる．ボクセル表現と違い, 高い解像度でも全ての点を求めず, オブジェクトの境界面のみ段階的に解像度を上げていくため, メモリ効率が非常によい．そうして高い解像度で得られた物体の境界部分を Marching cubes アルゴリズム [10] を用いて等値面を Mesh として抽出する．

### 3.2　データセットの作成

既存のデータセットにはカラー画像+深度画像の食事データセットは存在するが [4], 3D Mesh の食事のデータセットは存在しない．そこで本研究のため, 新しく食事の 3D データセットを作成した．今回作成したデータセットは食事の 3D モデル 252 個, 食器のモデル 38 個用意した．モデルの作成には, Structure Sensor 及び専用の 3D スキャンアプリケーションを使用した．異なる食事に対しても同じ食器を利用し撮影したため, 食事のモデルに対して食器のモデルが非常に少なくなっている．

#### 3.2.1　3D モデル

ネットワークの学習のためにはスキャンした 3D モデルはそのまま使用できない．主な問題点は以下の 3 つ．

(1) 原点中心にない
(2) 水密でない
(3) サイズが統一されていない
(4) ノイズが含まれている

まずスキャンしたモデルは原点ではないどこかに存在する．ネットワークの学習のためにはモデルは原点中心でなければならない．そこでまずモデルの中心を原点にし, 上向き+Y になるように移動と回転を施した．次にモデルが水密ではない問題について．本研究でベースにしている Occupancy Networks では, モデルの内側か外側かの確率を推論している．そのためモデルが水密モデルでないと, モデルの外側か内側かの定義ができないため訓練データを作成できない．そこでスキャンしたモデルたちを水密なモデルにする必要がある．穴が空いたモデルを埋めるアルゴリズムはいくつか存在する [1], [7], [8] が, 今回は Poisson Surface Reconstruction [7], [8] を用いた．しかしこのアルゴリズムをそのままモデルに適用はできない．そのままモデルに適応した場合の結果を図 3 に示した．モデルに Poisson Surface Reconstruction を適用した場合, モデルに存在する欠損がある程度小さい場合は図 3 の上の比較的綺麗に面が補完される．しかし図 3 の下のように, 期待した通りには面が作成されないモデルも多数存在した．Poisson Surface Reconstruction が期待するような面を作成しない主な理由として, 欠損が大きすぎるという点である．スキャンした 3D モデルは床との接地面が全て欠損している．そのため平皿のような床との接地面積が広いものは, 期待どおりに面が埋まらない．そこで Poisson Surface Reconstruction を適応する前に, 図 4 のように, ある程度接地面の穴を埋めるアルゴリズムを作成し適用することで穴を埋めた．アルゴリズムは至ってシンプルである．まず Y 軸についての最小値を持っている頂点を探索し, その一番下の頂点から Y 軸方向について一定の閾値以内 (今回は 0.002) の頂点全て取ってくる．この閾値以内の頂点を XZ 平面について $-\pi$〜$\pi$ のラジアン角を計算し, ソートする．

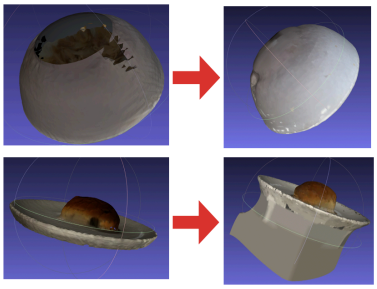

図 3　Poisson Surface Reconstruction の結果. 接地面の穴が小さいものはうまく補完されるが, 大きい場合にはうまくいかない

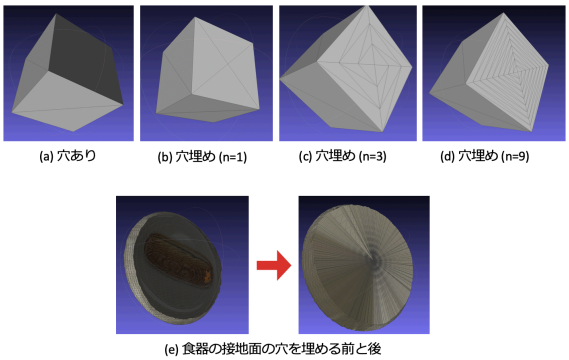


図 4　皿の接地面埋め

こうして得られたソート済みのデータ配列を用いて, 配列の i 番目と i+1 番目と原点を結ぶ 3 角面を作成することで床の面を埋めることが可能である. しかしただ埋めるだけでは不足である. なぜなら Poisson Surface Reconstruction で水密にする際, 各 3 角ポリゴンの大きさがあまりに違うとうまく再構成されないからである. なので原点と結ばれて作成された各 3 角面を $2n-1$ 個に分割する. 図 4 に n を増加させていった時, どのように分割されるかを示した. 以上の処理によって, モデルの表面の欠損を埋めた後, モデルのサイズを-0.5〜0.5 に正規化してサイズを統一する. 最後に図 5 のように, ノイズが残る問題が残る. このようなノイズに対して, TSDF Fusion を用いて再び Mesh を再構成することで対処した. TSDF Fusion とは Kinect Fusion [12] で提案された手法の一部を指す. Kinect Fusion は深度カメラのみを用いて SLAM と一定領域の Mesh の再構成手法である. 3D モデルを CG の仮想空間内で撮影する分には, カメラからみた物体の深度とカメラの位置は既知である. そのため Kinect Fusion の SLAM 部分を取り除き, Mesh の再構成のみを利用した形である. この手法を用いて, モデル内部にあるノイズを完全に取り除いた.

#### 3.2.2　画像

学習/検証用の画像のデータセットは各食事モデルを様々な角度からレンダリングして作成した. 各モデルに対して 25 枚の画像をレンダリングした. 図 6 にいくつかの例を示す.

### 3.3　学習

学習は 2 種類のネットワークで行った. 一つは食事の三次元形状のみを復元するネットワーク (Occupancy Networks). もう一つは食事と食器の 2 つの三次元形状を復元

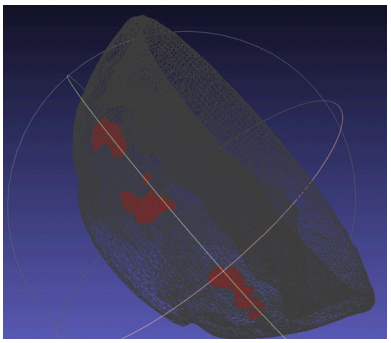

図 5　ノイズが含まれるモデル (赤い部分がノイズ)

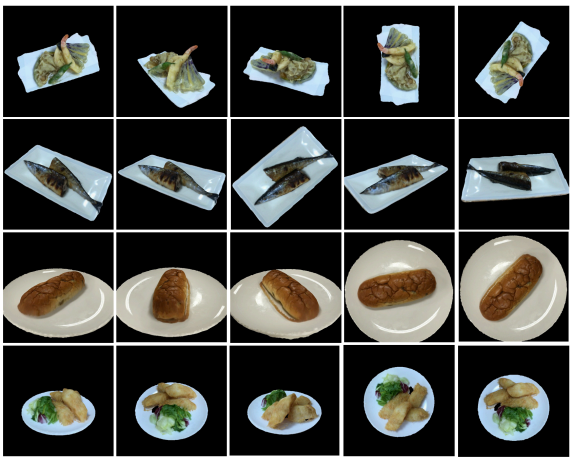

図 6　学習用にレンダリングした画像

するネットワークである (Hungry Networks). 双方共に学習には 228 モデル, テストに 24 モデルを使用した. 後者のモデルが前者に比べて精度がどのようになるかを比較するためである. まず前者の学習方法について説明する.

$p \in \mathbb{R}^3$ を入力の点, $x$ を入力画像とし, ネットワーク $f_\theta(p, x)$ のパラメータ $\theta$ を学習パラメータとする. ミニバッチの中の $i$ 番目のサンプルから, $xyz$ 軸それぞれ $-0.5$〜$0.5$ からなる空間からランダムに $K$ 個の点 $p_{ij} \in \mathbb{R}^3$ , $(j = 1, \ldots, K)$ を取ってくるとする時, ミニバッチ毎のロス $\mathcal{L}_\mathcal{B}$ を以下のように定める.

$$\mathcal{L}_\mathcal{B}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \sum_{j=1}^{K} \mathcal{L}(f_\theta(p_{ij}, x_i), o_{ij}) \tag{2}$$

ここで $x_i$ は バッチ $\mathcal{B}$ の $i$ 番目の画像であり, $o_{ij} \equiv o(p_{ij})$ は入力 $p_{ij}$ における答えの占有率である. また $\mathcal{L}(\cdot, \cdot)$ はクロスエントロピーロスである. これは Occupancy Networks と同じ損失関数である. 後者, つまり 3D モデルを 2 つ出力するネットワークでは, 上記の式を拡張し, 以下のような式を損失関数として用いた.

$$\mathcal{L}_\mathcal{B}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \Bigg( \sum_{j=1}^{K} \mathcal{L}(f_\theta(p1_{ij}, x_i), o1_{ij}) + \sum_{j=1}^{K} \mathcal{L}(f_\theta(p2_{ij}, x_i), o2_{ij}) \Bigg) \tag{3}$$

図 1 に示したように, decoder1,2 では入力される点は別々で

表 1　学習結果

| | batch size | IoU(食事) | IoU(食器) | 平均 IoU |
|---|---|---|---|---|
| 食事のみ | 32 | 0.6287 | | 0.6287 |
| 食事と食器 | 32 | 0.3887 | 0.3214 | 0.3550 |
| 食事と食器 | 128 | 0.5902 | 0.8163 | 0.7033 |

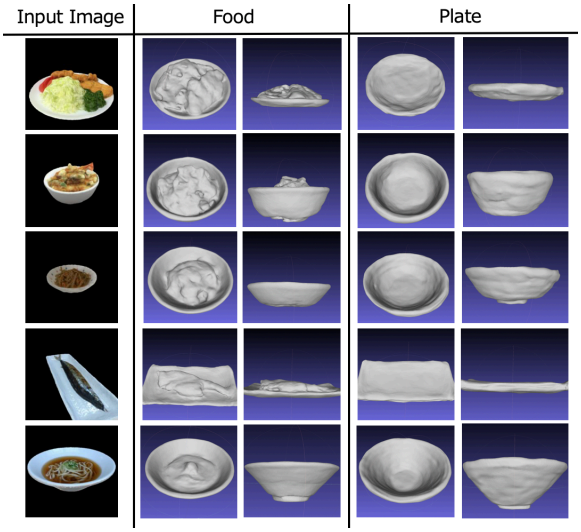


図 7　単一画像からの Mesh の生成の結果

あり, それぞれ $p1, p2 \in \mathbb{R}^3$ とする. また占有率も当然モデルごとによって異なり, $p1, p2$ に対応するものを $o1, o2 \in \mathbb{R}$ とする. 学習はテストデータで IoU の精度が上がらなくなるまで学習した.

## 4. 結果

ネットワークの評価には IoU を用いた. 3D モデルの IoU の計算にはモデルのバウンディングの中から 10 万点一様にサンプリングし, それら物体内部として推論された結果と正しい結果の積集合と和集合の計算することでわかる. その結果を表 1 に示す. はじめに食器のモデルを使用せず, 食事のモデルのみを用いて学習したところ, IoU が 0.6287 であった. そこでバッチサイズは変えずに decoder を 2 つ増やしたネットワークで食事と食器の両方のモデルを生成するよう学習したところ, まったく精度が出なかった. そこでバッチサイズを 128 にしたところ, 食事のモデルの IoU は食事のみの場合と比べて若干落ちたものの, 平均 IoU が 0.7 程度が出た. 図 7 に入力画像と生成結果を示した.

## 5. おわりに

本研究では食事の体積推定のため, 3D Mesh の食事データセットを作成し, 単一食事画像から食事と食器の三次元形状復元を実現する Hungry Networks を作成し, 平均 IoU0.7 程度と高精度に実現した. 今後の課題としては, レンダリング画像ではなく, 現実の背景や隣接してが複雑な食事画像から食事と食器の三次元形状復元を行い, カロリー量推定などに結びつけたいと考えている.

## 参考文献

[1] Calakli, F. and Taubin, G.: SSD: Smooth signed distance surface reconstruction, *Computer Graphics Forum*, Vol. 30, No. 7, pp. 1993–2002 (2011).

[2] Choy, C. B., Xu, D., Gwak, J., Chen, K. and Savarese, S.: 3D-R2N2: A unified approach for single and multi-view 3d object reconstruction, *Proc.of European Conference on Computer Vision*, pp. 628–644 (2016).

[3] Fan, H., Su, H. and Guibas, L. J.: A Point Set Generation Network for 3D Object Reconstruction from a Single Image, *Proc.of IEEE Computer Vision and Pattern Recognition*, pp. 605–613 (2017).

[4] Ferdinand, C. P., Schlecht, S., Ettlinger, F., Grun, F., Heinle, C., Tatavatry, S., Ahmadi, S. A., Diepold, K. and Menze, B. H.: Diabetes60-Inferring Bread Units From Food Images Using Fully Convolutional Neural Networks, *Proc.of the IEEE International Conference on Computer Vision Workshops*, pp. 1526–1535 (2017).

[5] Gkioxari, G., Malik, J. and Johnson, J.: Mesh R-CNN, *Proc.of IEEE International Conference on Computer Vision*, pp. 9785–9795 (2019).

[6] Kanazawa, A., Tulsiani, S., Efros, A. A. and Malik, J.: Learning category-specific mesh reconstruction from image collections, *Proc.of European Conference on Computer Vision*, pp. 371–386 (2018).

[7] Kazhdan, M., Bolitho, M. and Hoppe, H.: Poisson surface reconstruction, *Proc.of the fourth Eurographics symposium on Geometry processing*, Vol. 7 (2006).

[8] Kazhdan, M. and Hoppe, H.: Screened poisson surface reconstruction, *ACM Transactions on Graphics (ToG)*, Vol. 32, No. 3, pp. 1–13 (2013).

[9] Lin, M., Chen, Q. and Yan, S.: Network in network, *arXiv preprint arXiv:1312.4400* (2013).

[10] Lorensen, W. E. and Cline, H. E.: Marching cubes: A high resolution 3D surface construction algorithm, *ACM siggraph computer graphics*, Vol. 21, No. 4, pp. 163–169 (1987).

[11] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S. and Geiger, A.: Occupancy Networks: Learning 3d reconstruction in function space, *Proc.of IEEE Computer Vision and Pattern Recognition*, pp. 4460–4470 (2019).

[12] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S. and Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking, *Proc.of 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136 (2011).

[13] Ranjan, A., Bolkart, T., Sanyal, S. and Black, M. J.: Generating 3D faces using convolutional mesh autoencoders, *Proc.of European Conference on Computer Vision*, pp. 704–720 (2018).

[14] Tulsiani, S., Zhou, T., Efros, A. A. and Malik, J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency, *Proc.of IEEE Computer Vision and Pattern Recognition*, pp. 2626–2634 (2017).

[15] Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W. and Jiang, Y. G.: Pixel2mesh: Generating 3d mesh models from single rgb images, *Proc.of European Conference on Computer Vision*, pp. 52–67 (2018).

[16] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X. and Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes, *Proc.of IEEE Computer Vision and Pattern Recognition*, pp. 1912–1920 (2015).