

陰関数表現とRGB-D画像を用いた 食事と食器の実寸三次元再構成

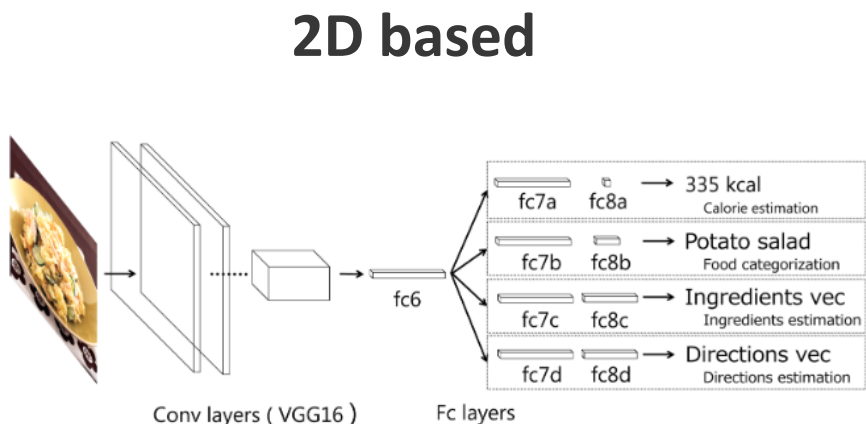
成富志優 柳井啓司

電気通信大学 大学院 情報学専攻

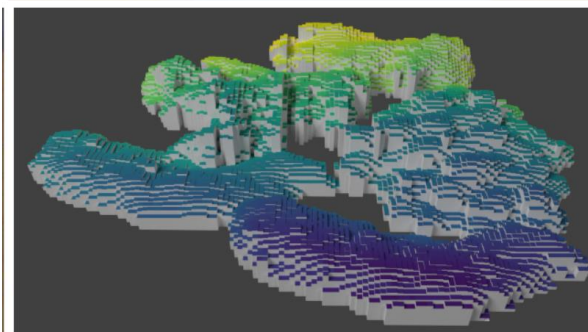
はじめに

- 食事のカロリー量/栄養管理などは重要なトピック。
- 様々な手法/アプリケーションが研究開発されている。
 - しかし多くが**平面的認識**、或いは食事は**平皿上**にあるという前提。

Depth based

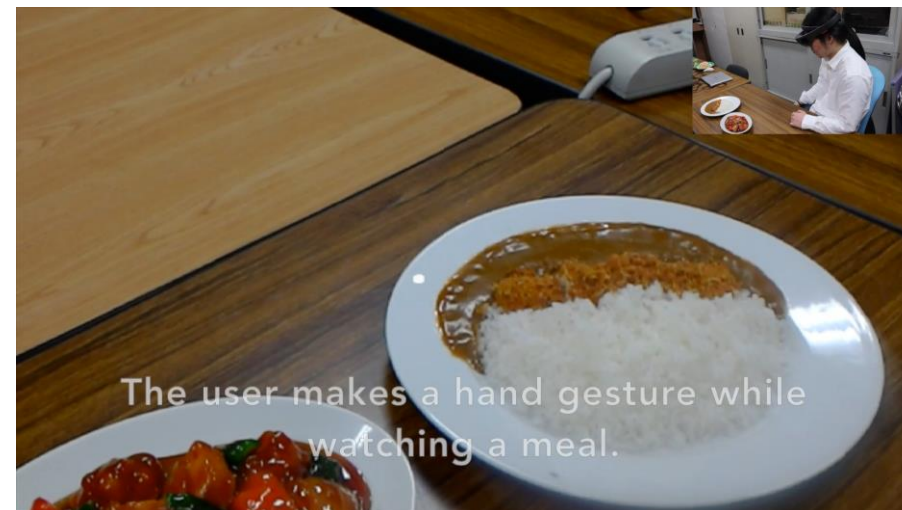


[Ege et al., IEICE2018]



[Im2Calories, ICCV 2015]

Sensor based



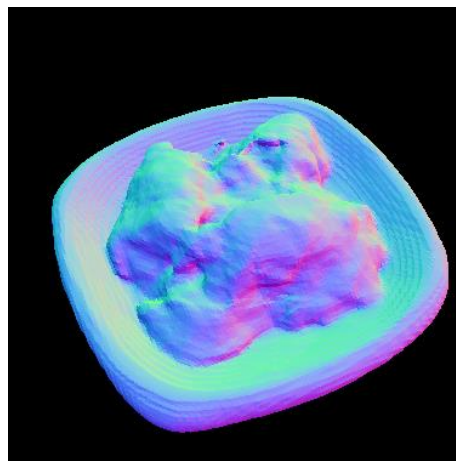
[CalorieCaptorGlass, IEEE VR 2020]

研究の目的

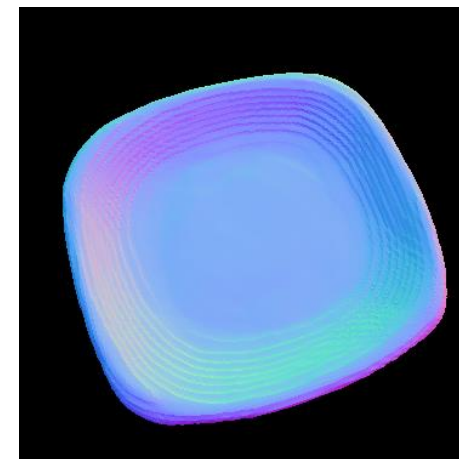
- 食事を**立体的**に認識するため、**三次元再構成**を行う。



食事画像



食事



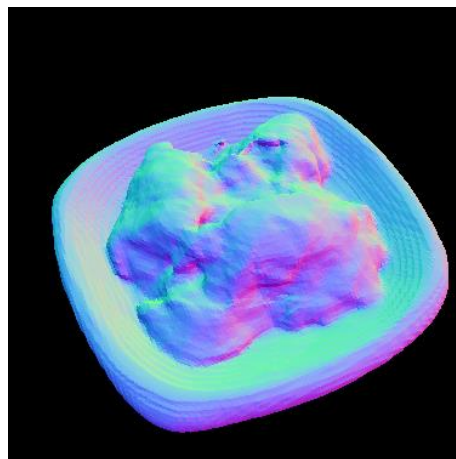
食器

- 食事を**立体的**に認識するため、**三次元再構成**を行う。

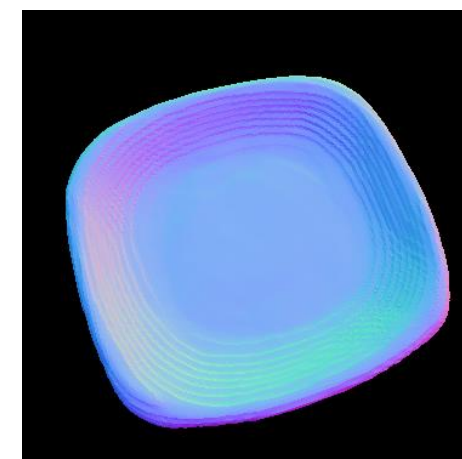
食事と**食器**を同時に再構成



食事画像



食事



食器

研究の目的

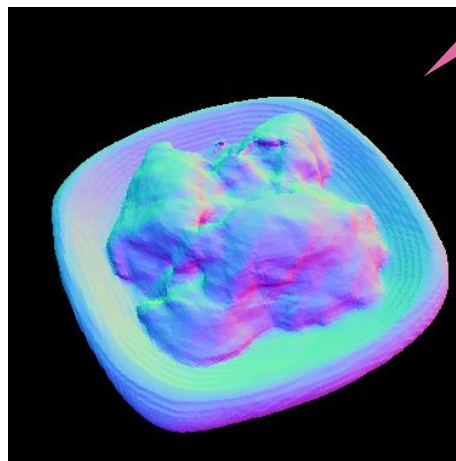
- 食事を**立体的**に認識するため、**三次元再構成**を行う。

食事と**食器**を同時に再構成

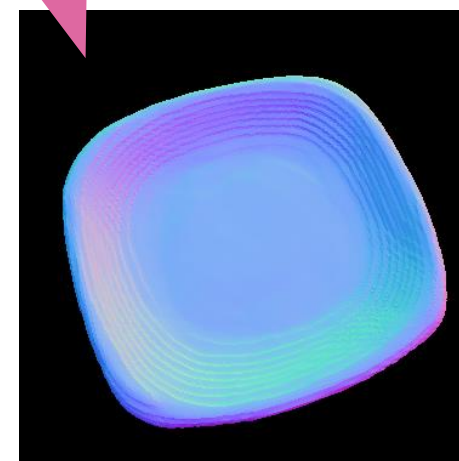
双方の体積の差分から
食品の体積を計算



食事画像



食事

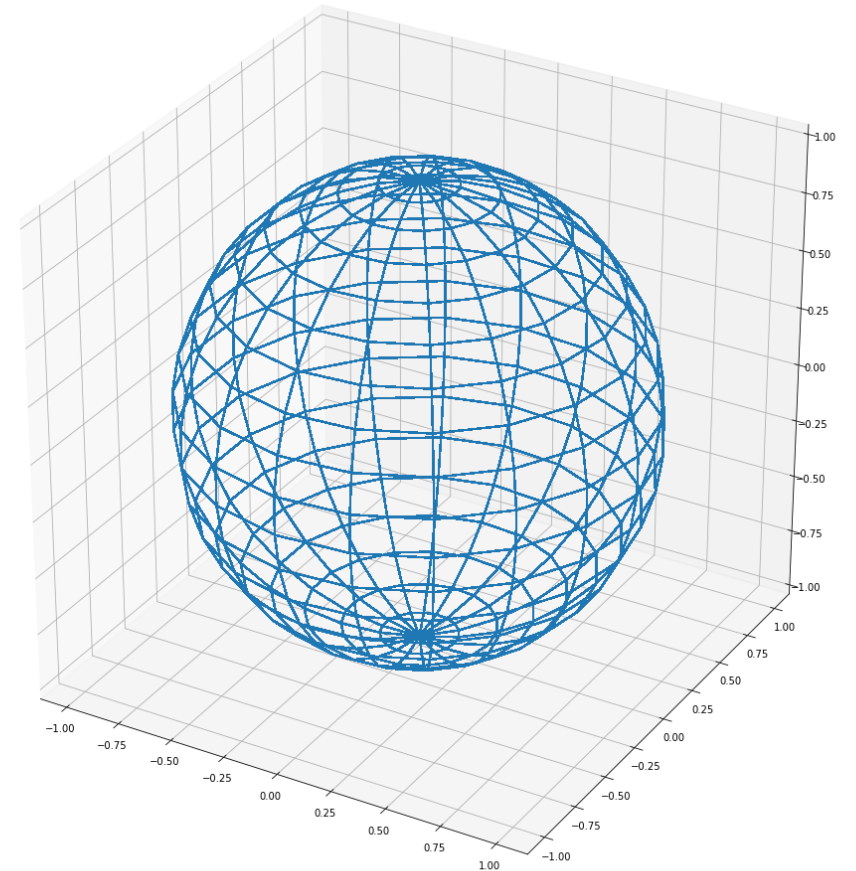


食器

陰関数表現

陰関数表現とは

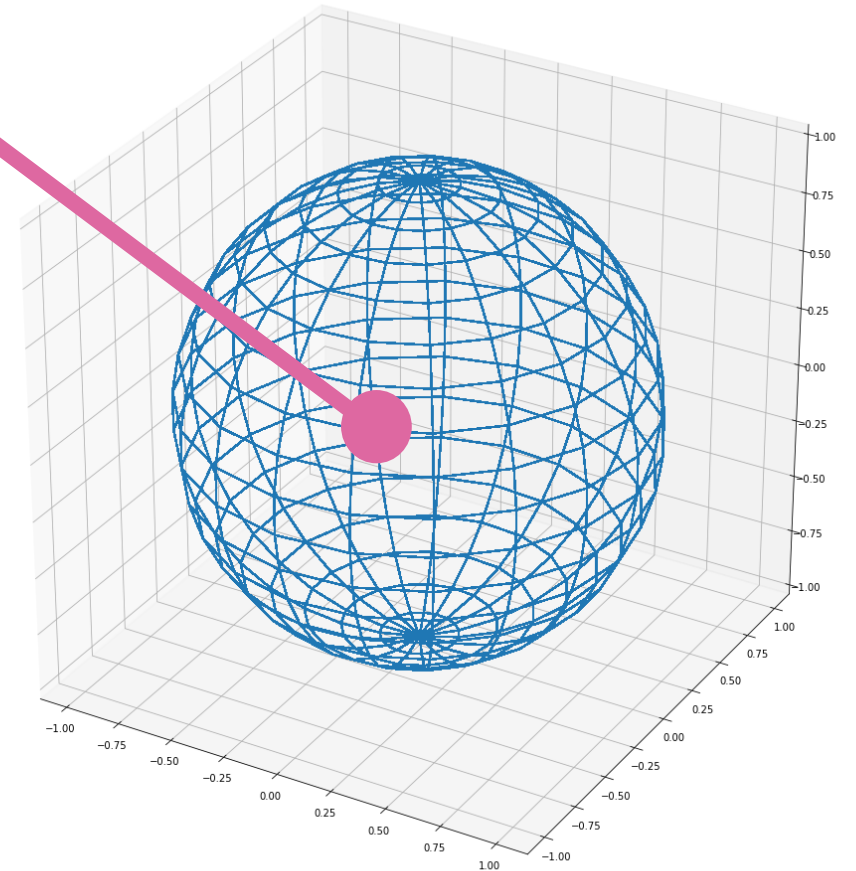
$$f(x, y, z) = \begin{cases} 1 & \text{if } x^2 + y^2 + z^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



単位球

陰関数表現とは

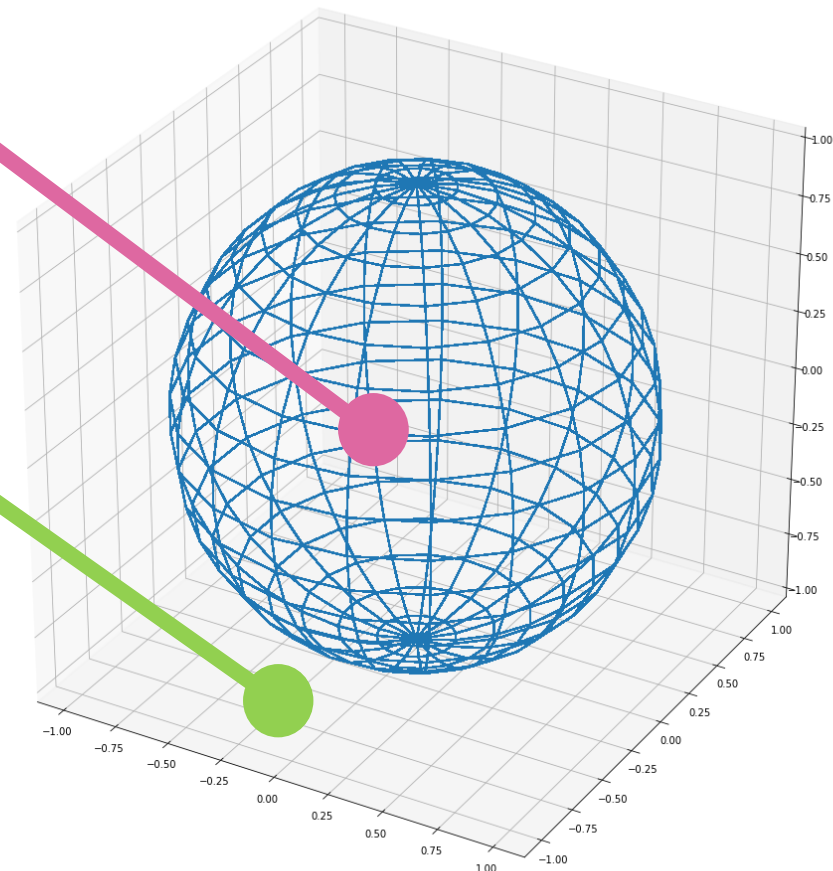
$$f(x, y, z) = \begin{cases} 1 & \text{if } x^2 + y^2 + z^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



単位球

陰関数表現とは

$$f(x, y, z) = \begin{cases} 1 & \text{if } x^2 + y^2 + z^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

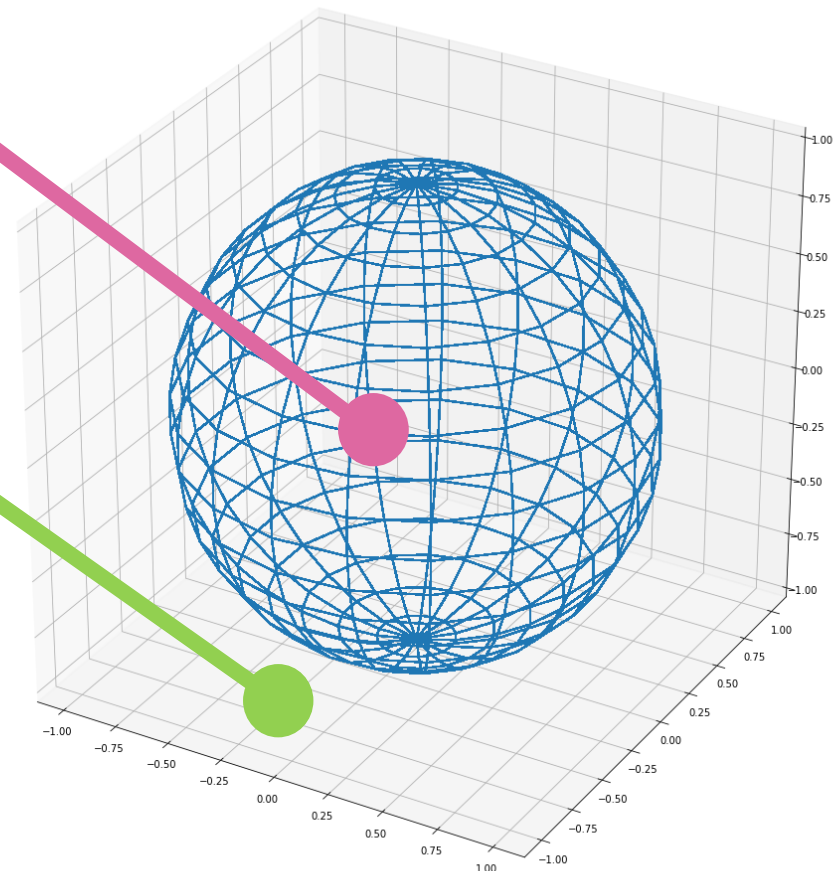


単位球

陰関数表現とは

$$f(x, y, z) = \begin{cases} 1 & \text{if } x^2 + y^2 + z^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

占有率
in/out



単位球

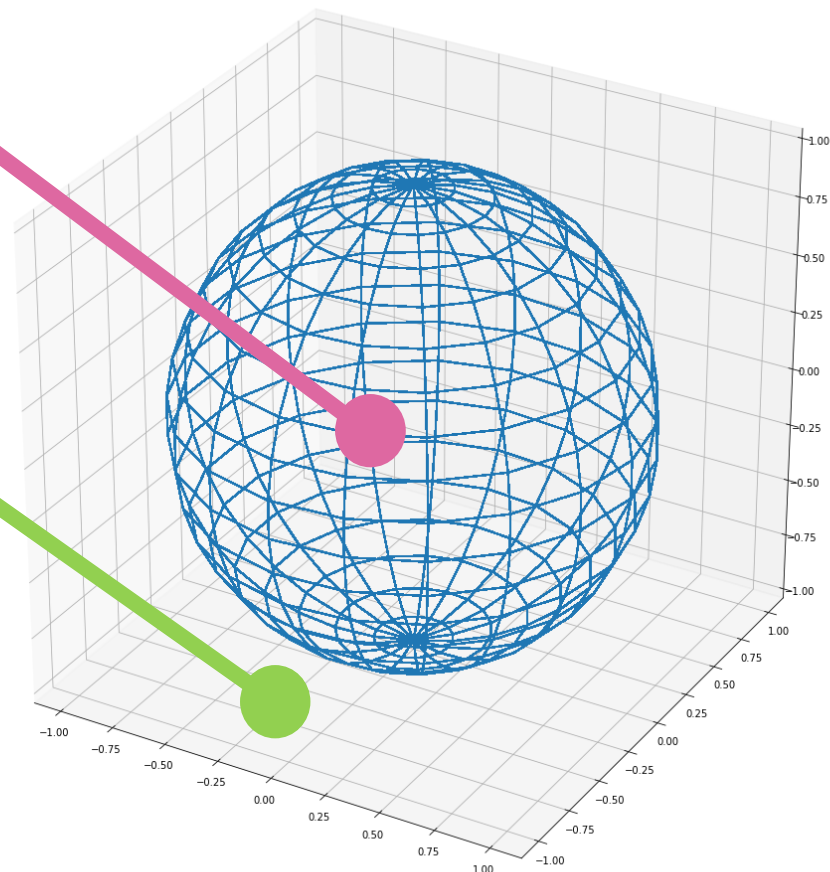
陰関数表現とは

$$f(x, y, z) = \begin{cases} 1 & \text{if } x^2 + y^2 + z^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

占有率
in/out

$dnn(x, y, z | X)$ X : 特徴量

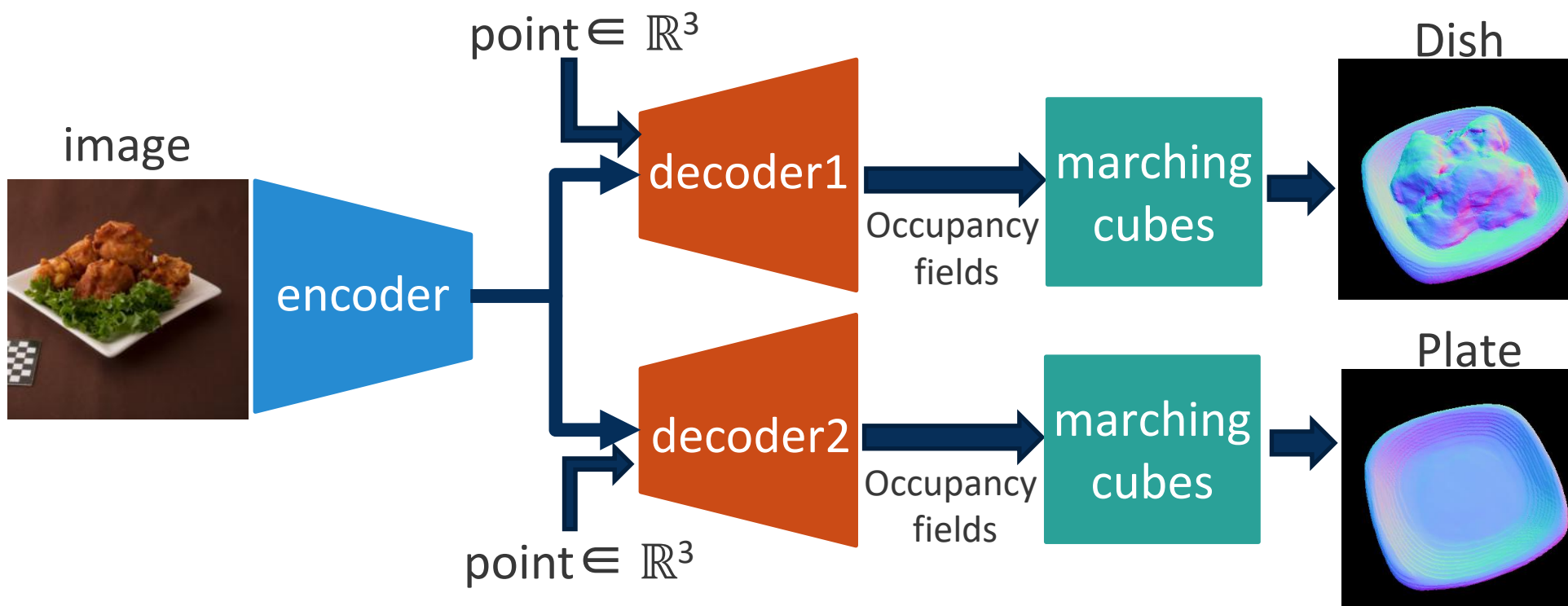
学習対象



単位球

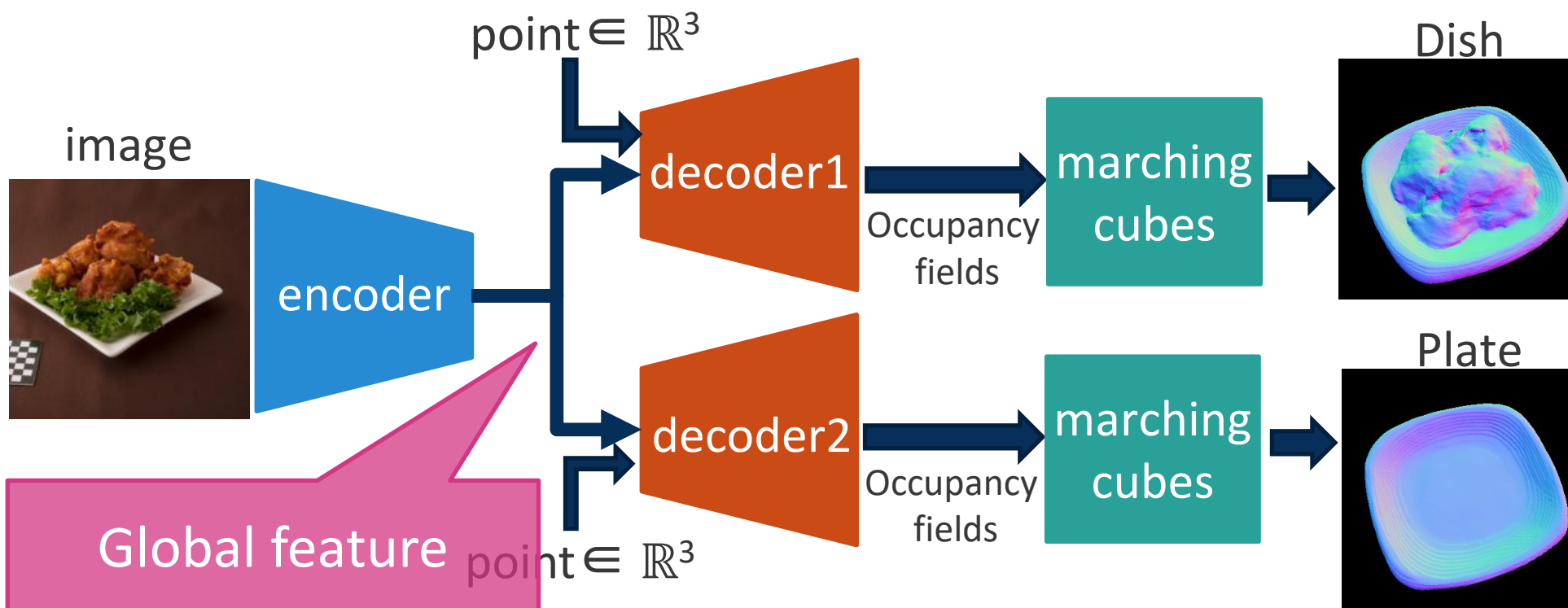
過去の研究: Hungry Networks[1]

- 単一の **RGB** 画像を入力に**食事と食器の三次元形状を高解像度に再構成する陰関数表現**を用いた手法。



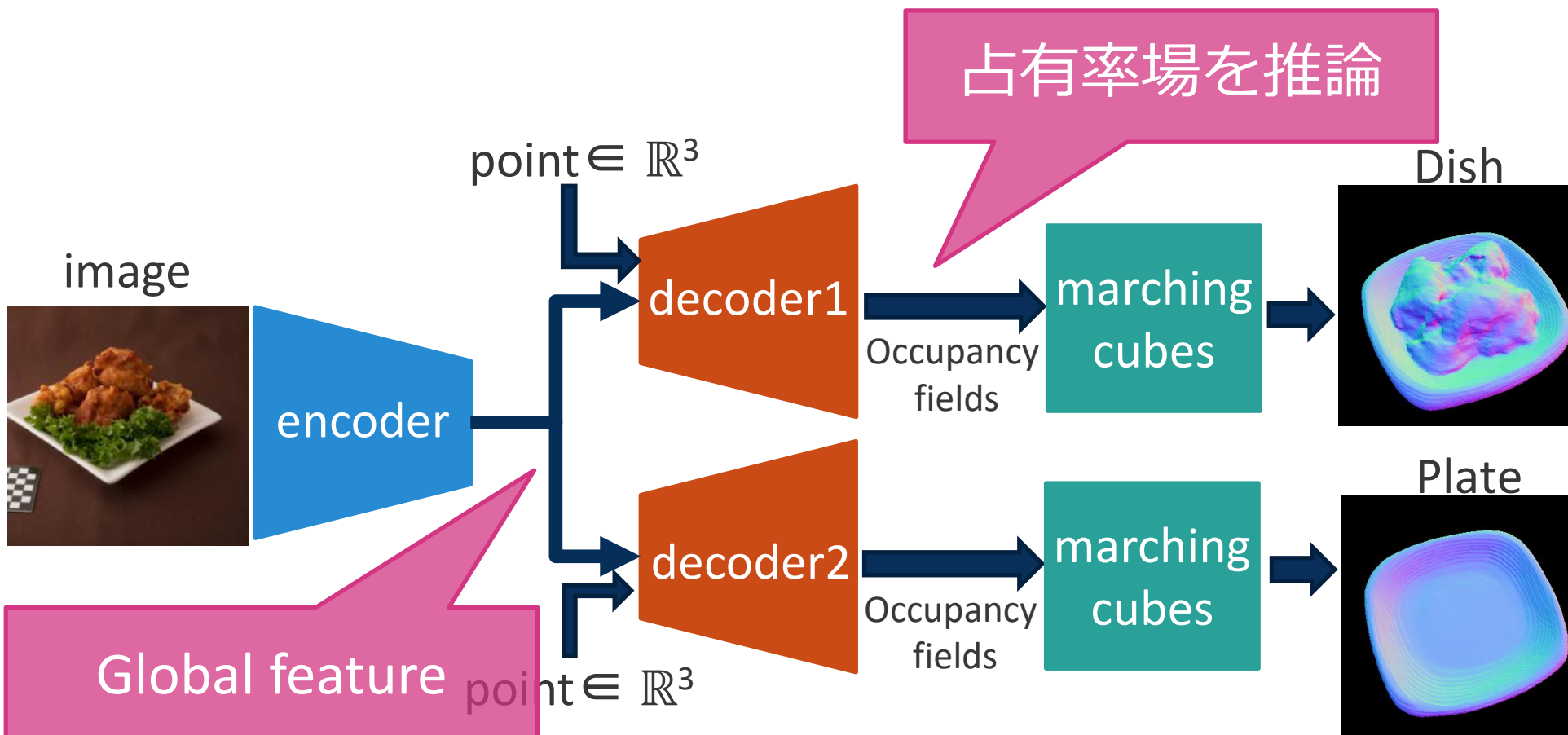
過去の研究: Hungry Networks[1]

- 単一の **RGB** 画像を入力に**食事と食器の三次元形状を高解像度に再構成する陰関数表現**を用いた手法。



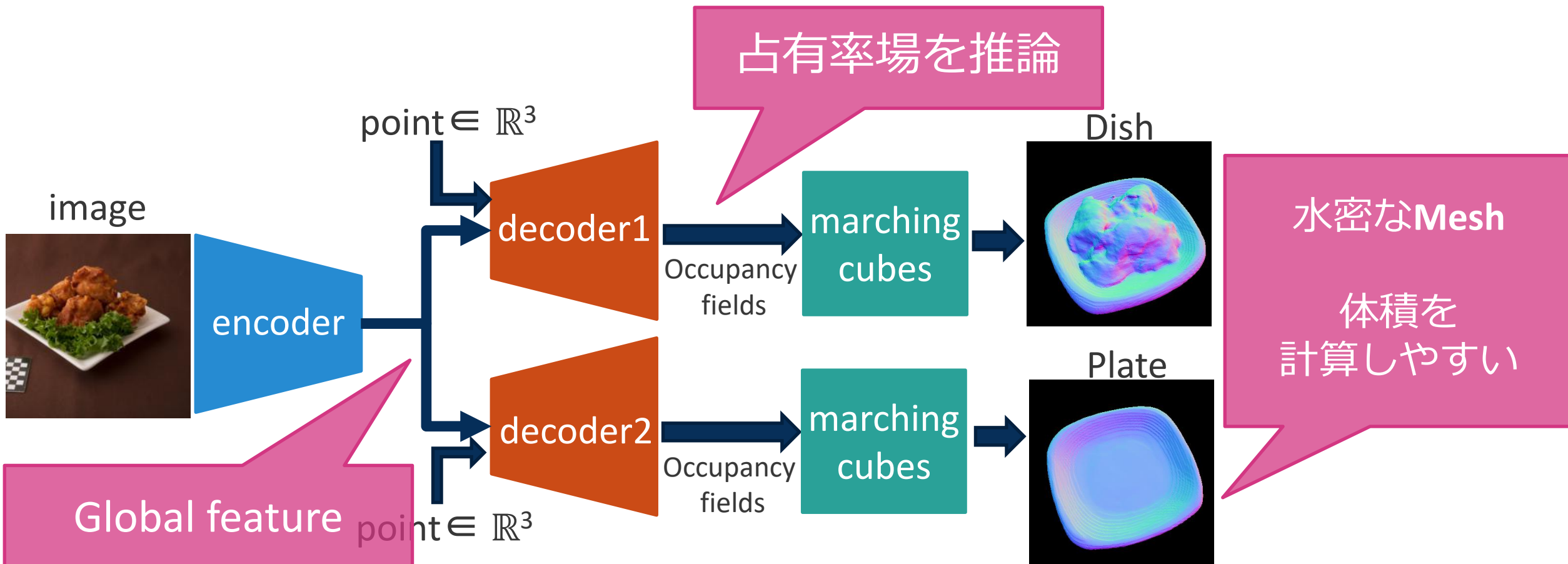
過去の研究: Hungry Networks[1]

- 単一の **RGB** 画像を入力に**食事と食器の三次元形状を高解像度に再構成する陰関数表現**を用いた手法。



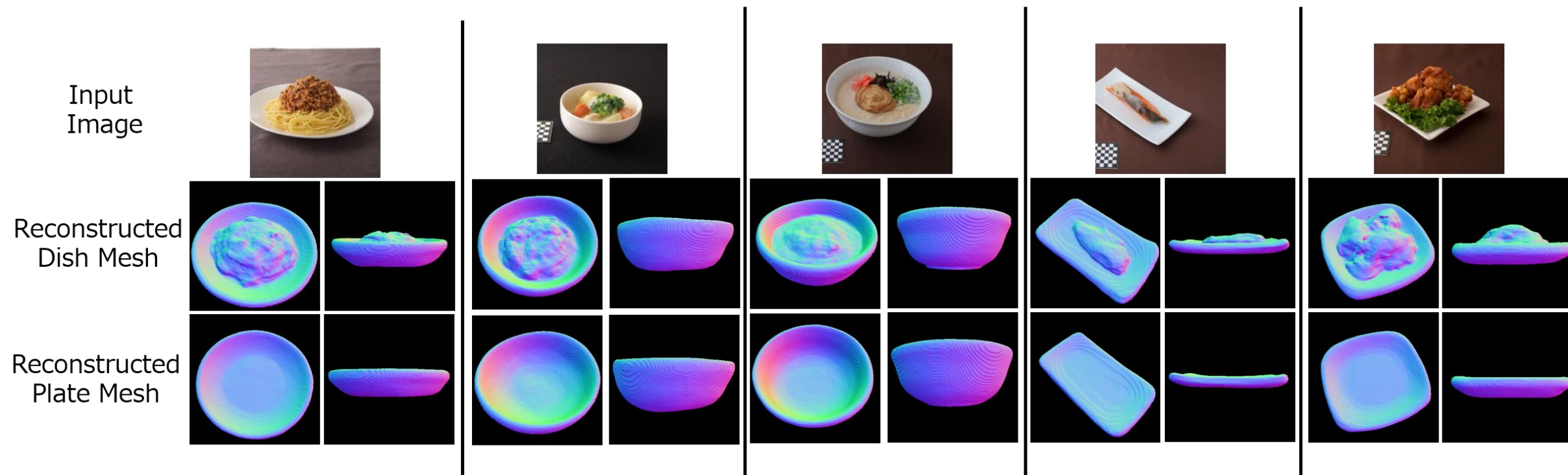
過去の研究: Hungry Networks[1]

- 単一の **RGB** 画像を入力に**食事と食器の三次元形状を高解像度**に再構成する**陰関数表現**を用いた手法。

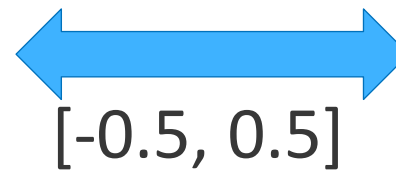
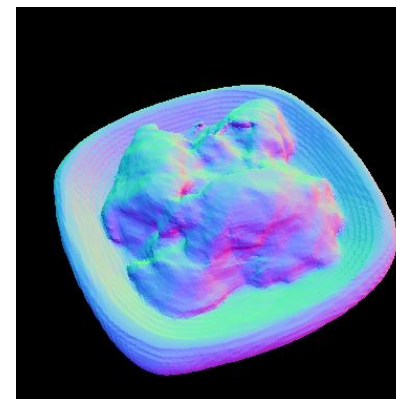


過去の研究: Hungry Networks[1]

- 高精度な食事と食器の**再構成/体積推定**を実現。

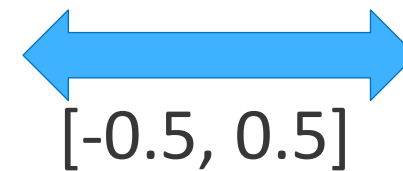
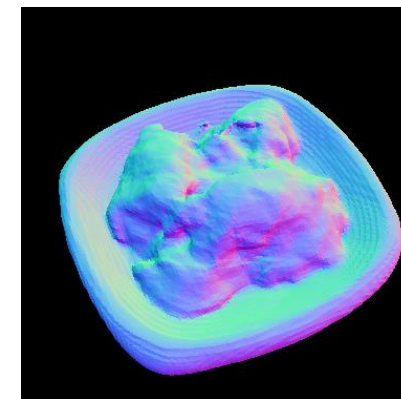


- 再構成された三次元形状は**正規化**されている。



Hungry Networks の課題

- 再構成された三次元形状は**正規化**されている。



正規化されているため
学習がうまくいく。

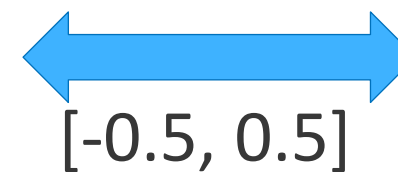
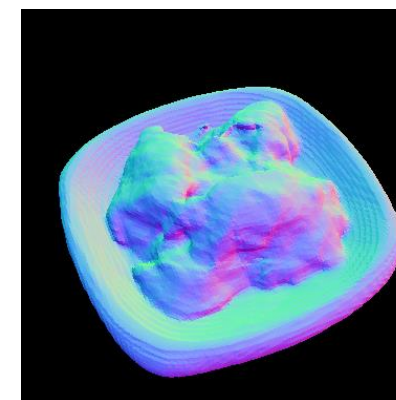
Hungry Networks の課題

- 再構成された三次元形状は**正規化**されている。

食品の実際の体積を知りたい。



何らかの手段で**実寸**を
計測する必要がある。



正規化されているため
学習がうまくいく。

陰関数表現 + 単一 RGB-D 画像
= 実寸三次元再構成

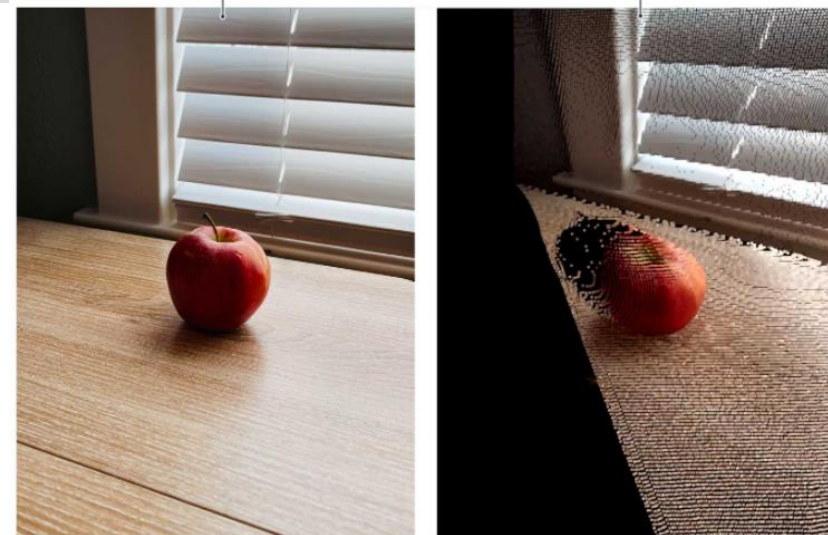
カメラモデル + 深度画像 → 実寸

カメラモデル + 深度画像 → 実寸



[Apple 公式サイトから引用]

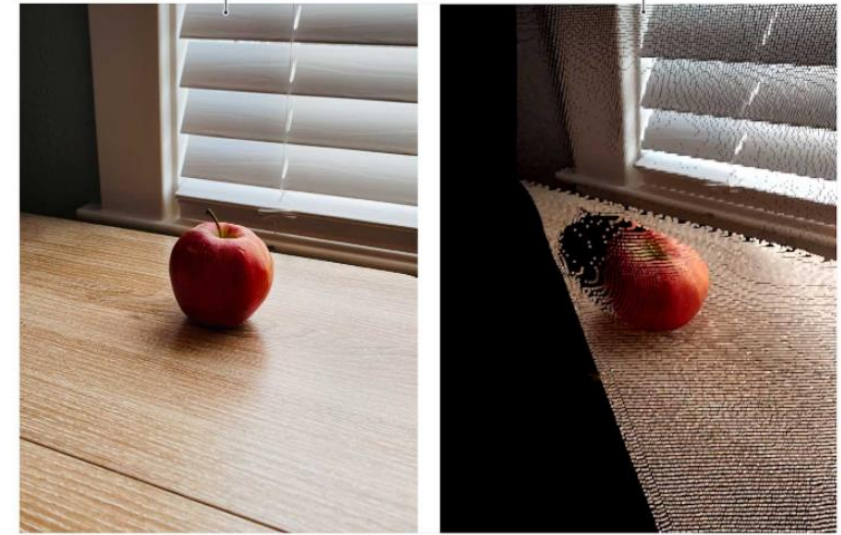
カメラモデル + 深度画像 → 実寸



[Apple 公式サイトから引用]

体積推定 → 完全な三次元形状を再構成する必要性

カメラモデル + 深度画像 → 実寸



[Apple 公式サイトから引用]

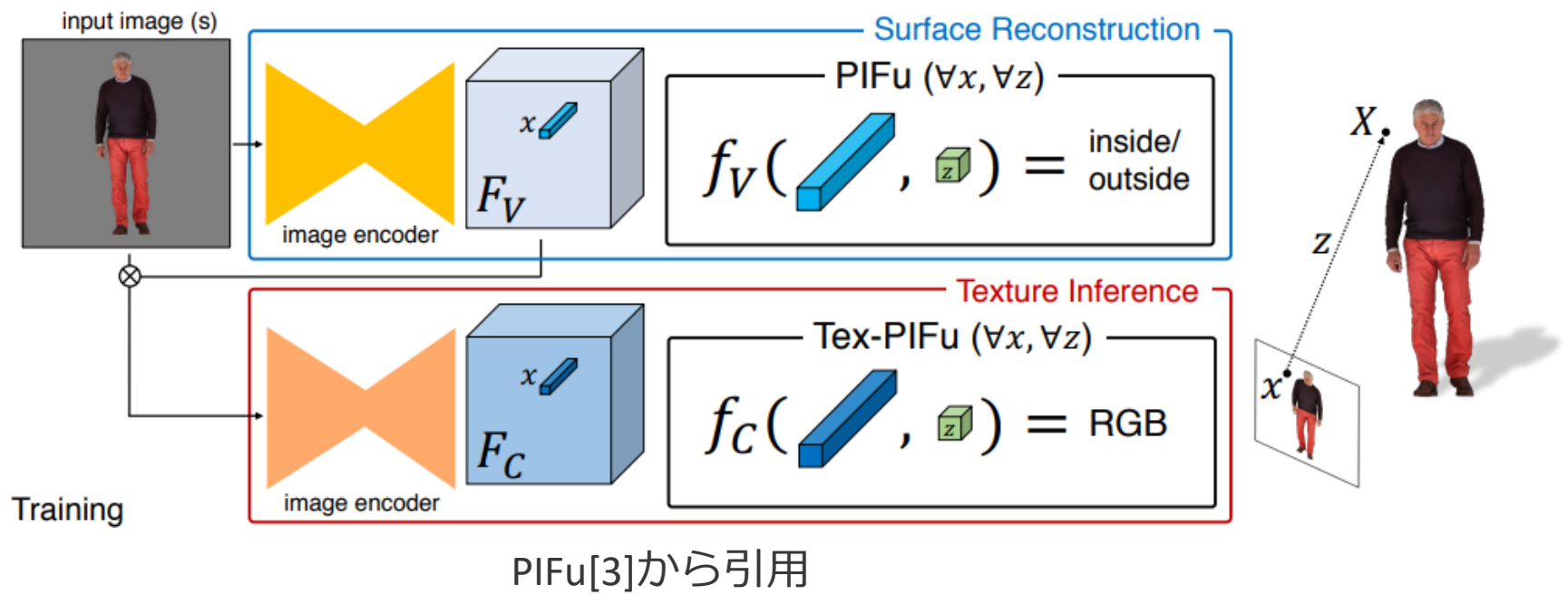
体積推定 → 完全な三次元形状を再構成する必要性



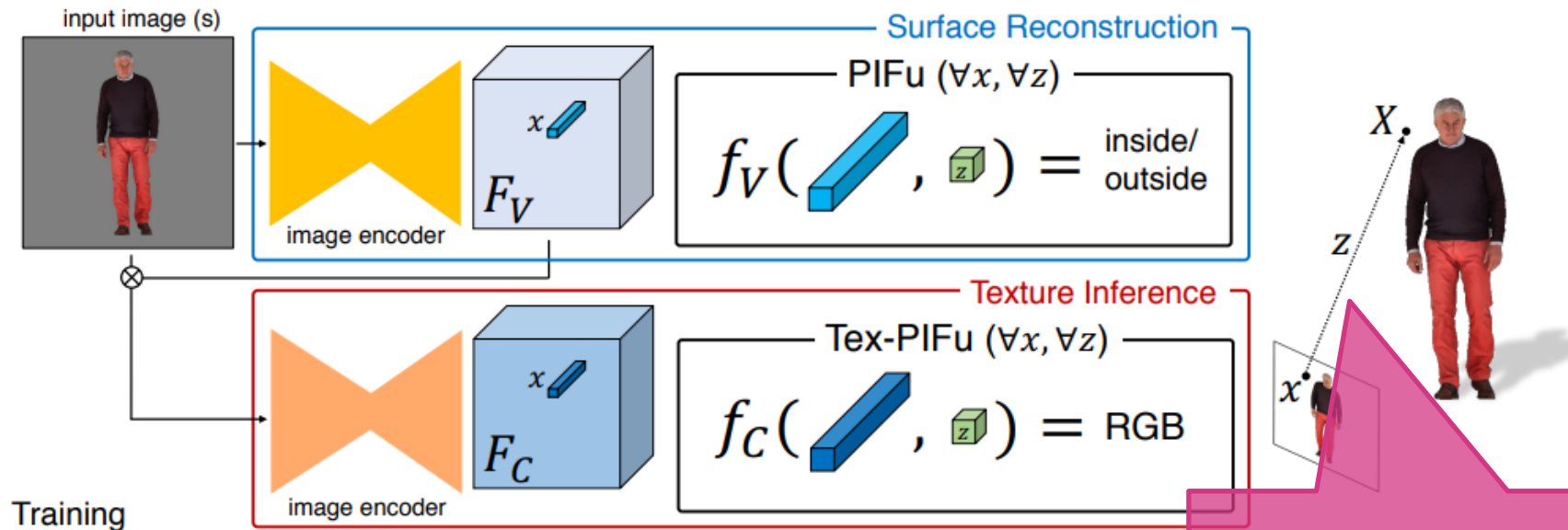
カメラモデル + 深度画像 + 三次元再構成を

どのように統合するか。

- PIFu[3]: RGB画像から人の三次元再構成



- PIFu[3]: RGB画像から人の三次元再構成

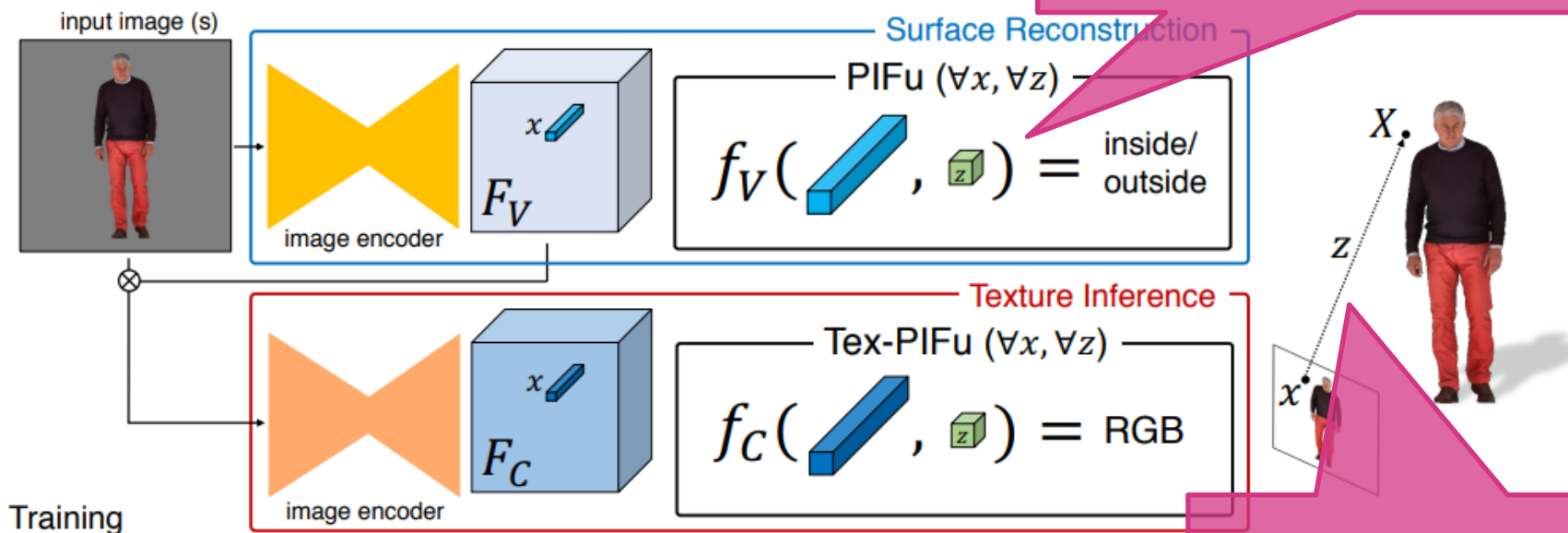


PIFu[3]から引用

世界座標上の $X \in R^3$ と画像座標上の $x \in R^2$ の対応を計算。

- PIFu[3]: RGB画像から人の三次元再構成

画像座標 x に対応する
特徴量と深度を入力に
占有率を推論

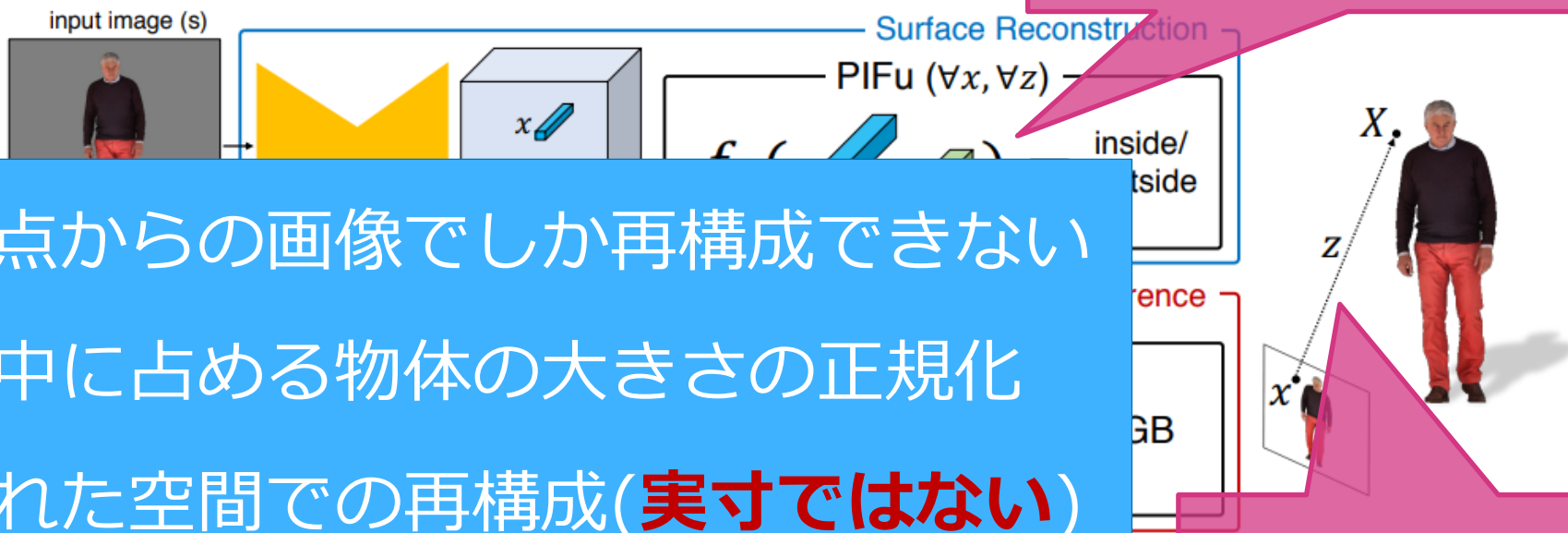


PIFu[3]から引用

世界座標上の $X \in R^3$ と
画像座標上の $x \in R^2$ の
対応を計算。

- PIFu[3]: RGB画像から人の三次元再構成

画像座標 x に対応する
特徴量と深度を入力に
占有率を推論

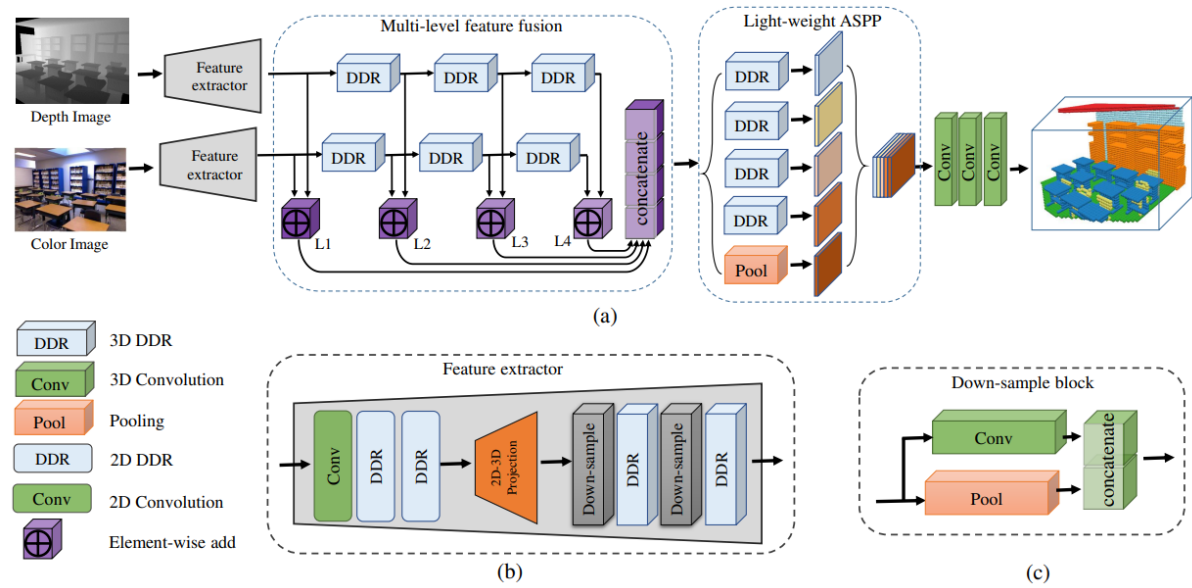


- ①特定の視点からの画像でしか再構成できない
- ②入力画像中に占める物体の大きさの正規化
- ③正規化された空間での再構成(実寸ではない)

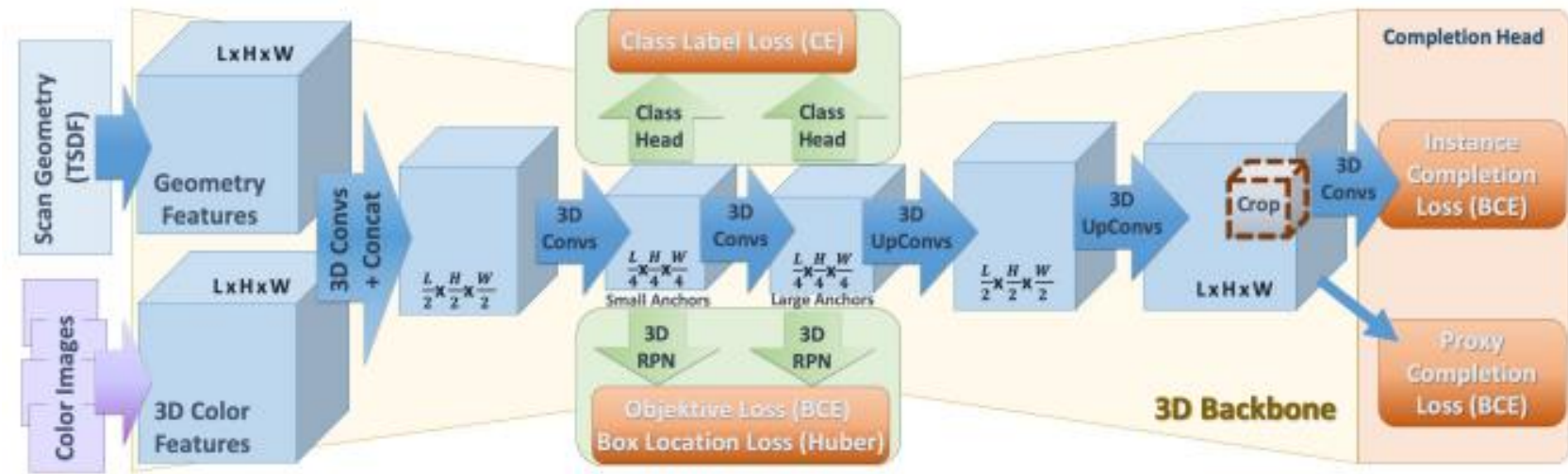
PIFu[3]から引用

世界座標上の $X \in R^3$ と
画像座標上の $x \in R^2$ の
対応を計算。

- RGB-D画像を用いた三次元再構成



Liらの手法[4]から引用



Houらの手法[5]から引用

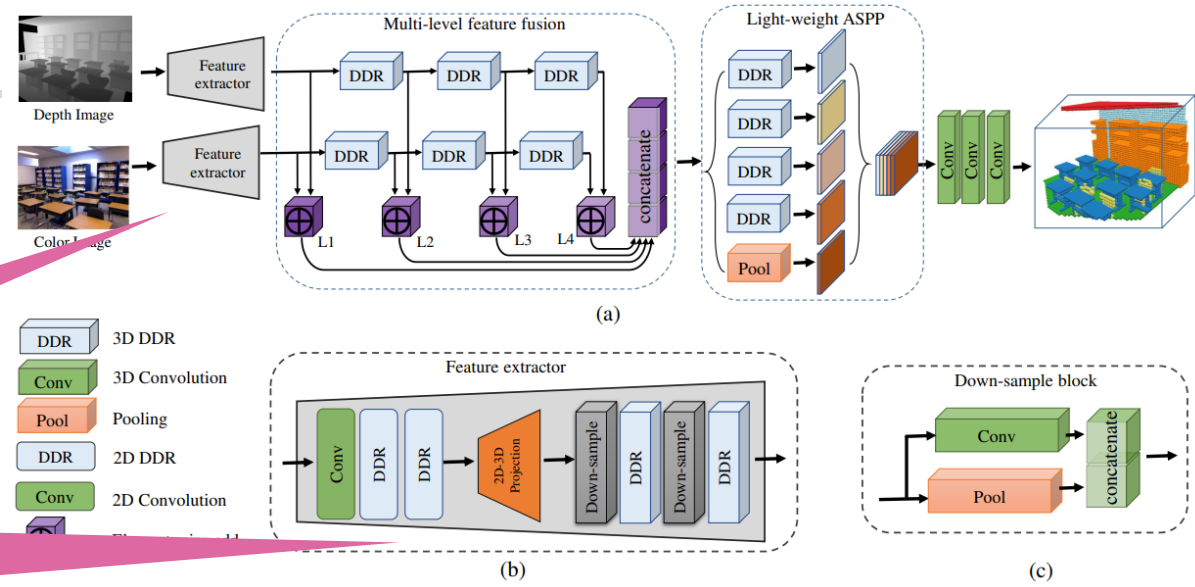
[4] RGBD Based Dimensional Decomposition Residual Network for 3D Semantic Scene Completion, CVPR 2019

[5] RevealNet: Seeing Behind Objects in RGB-D Scans, CVPR 2020

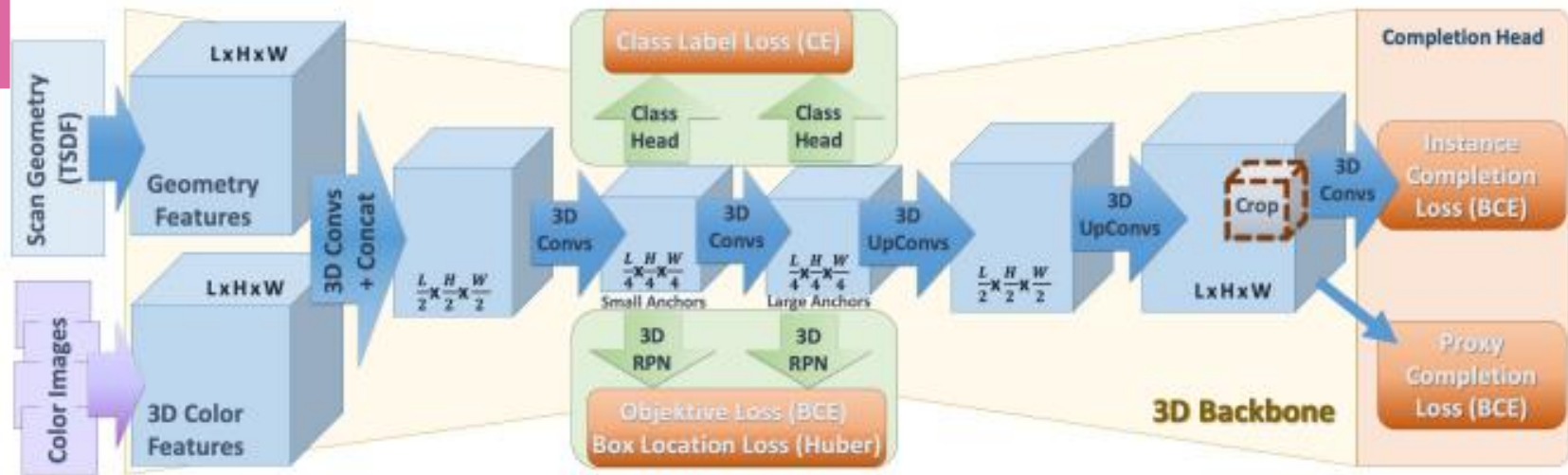
関連研究

- RGB-D画像を用いた三次元再構成

3D Gridに画像特徴量を
Back projection



Liらの手法[4]から引用



Houらの手法[5]から引用

[4] RGBD Based Dimensional Decomposition Residual Network for 3D Semantic Scene Completion, CVPR 2019

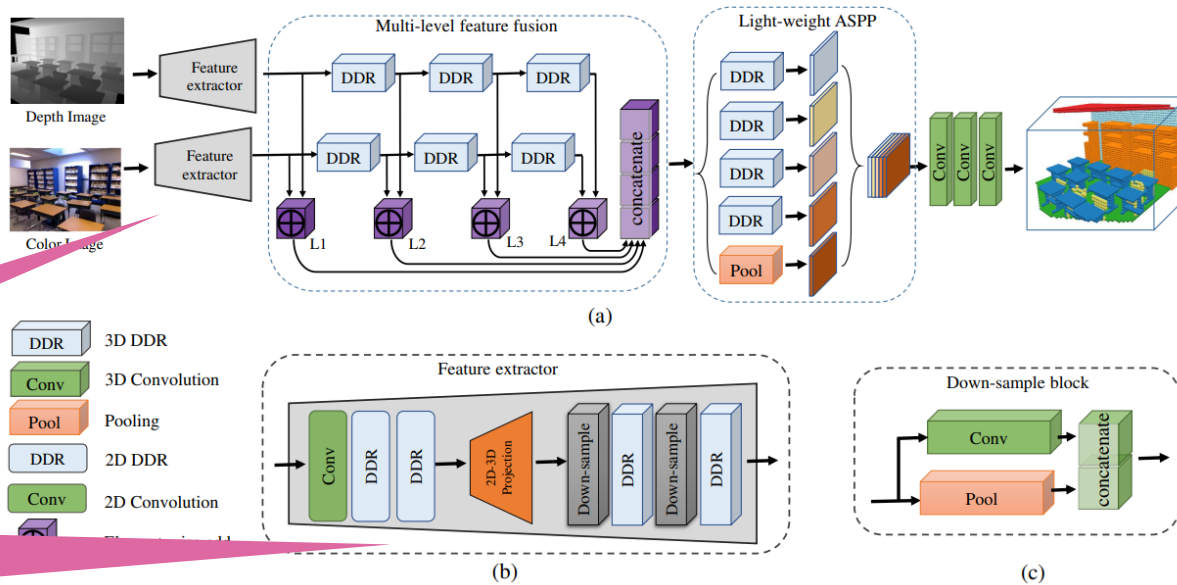
[5] RevealNet: Seeing Behind Objects in RGB-D Scans, CVPR 2020

関連研究

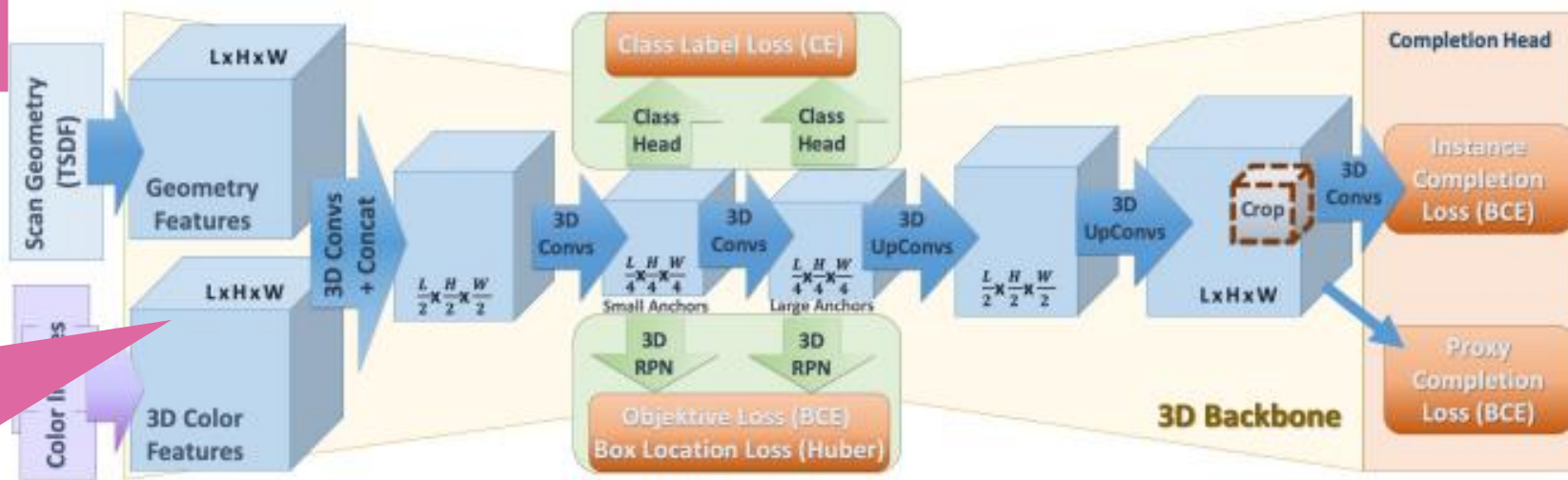
- RGB-D画像を用いた三次元再構成

3D Gridに画像特徴量を
Back projection

Depthを
TSDF(3D Grid)に変換



Liらの手法[4]から引用



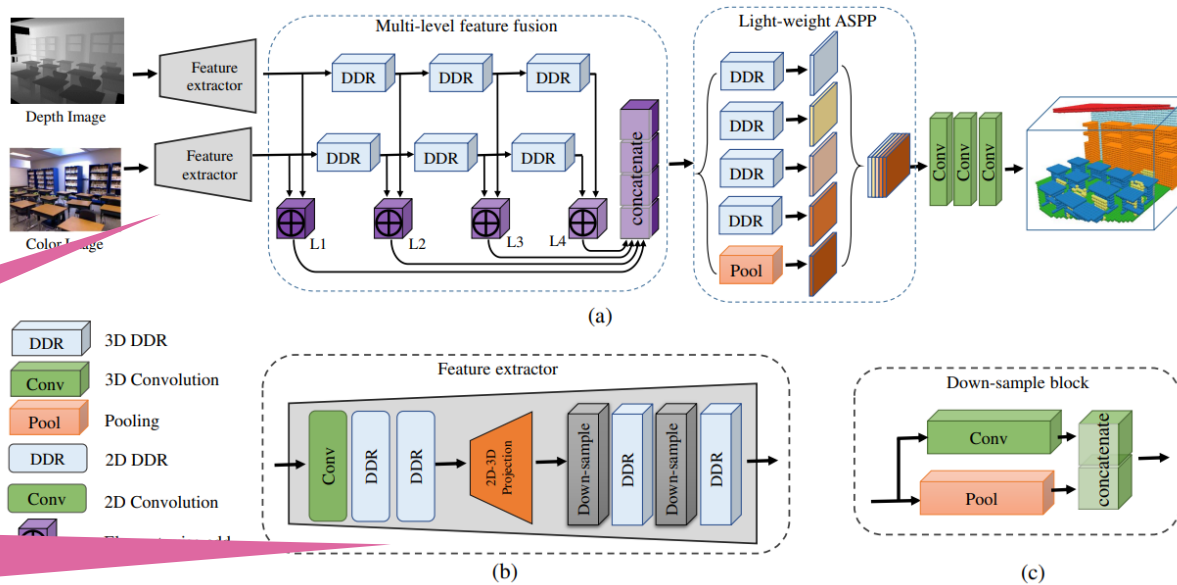
Houらの手法[5]から引用

[4] RGBD Based Dimensional Decomposition Residual Network for 3D Semantic Scene Completion, CVPR 2019

[5] RevealNet: Seeing Behind Objects in RGB-D Scans, CVPR 2020

関連研究

- RGB-D画像を用いた三次元再構成



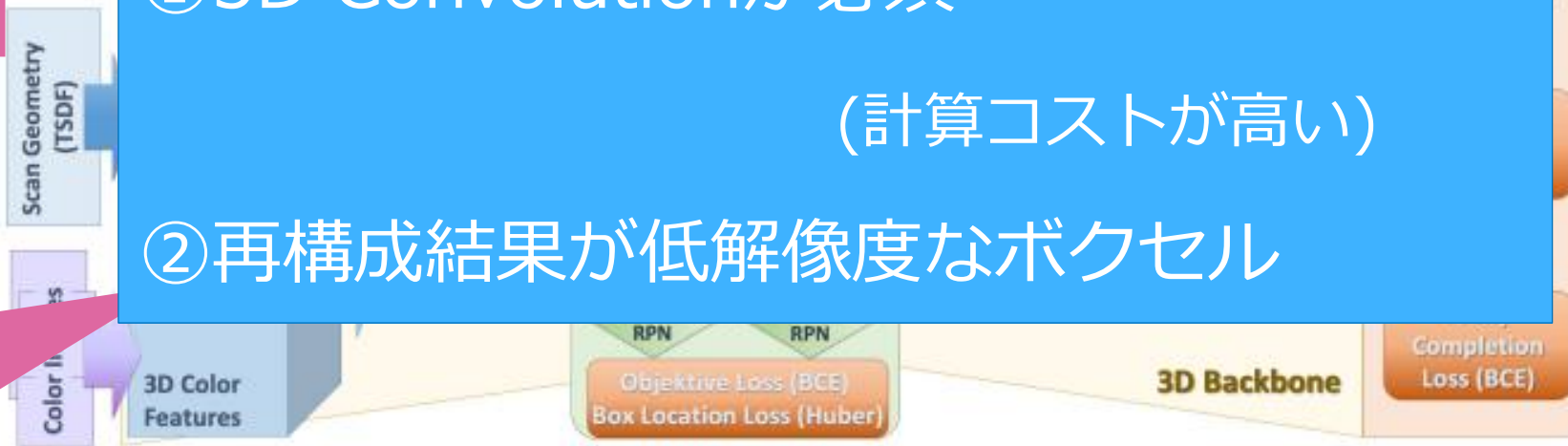
3D Gridに画像特徴量を
Back projection

① 3D Convolutionが必須

(計算コストが高い)

② 再構成結果が低解像度なボクセル

Depthを
TSDF(3D Grid)に変換

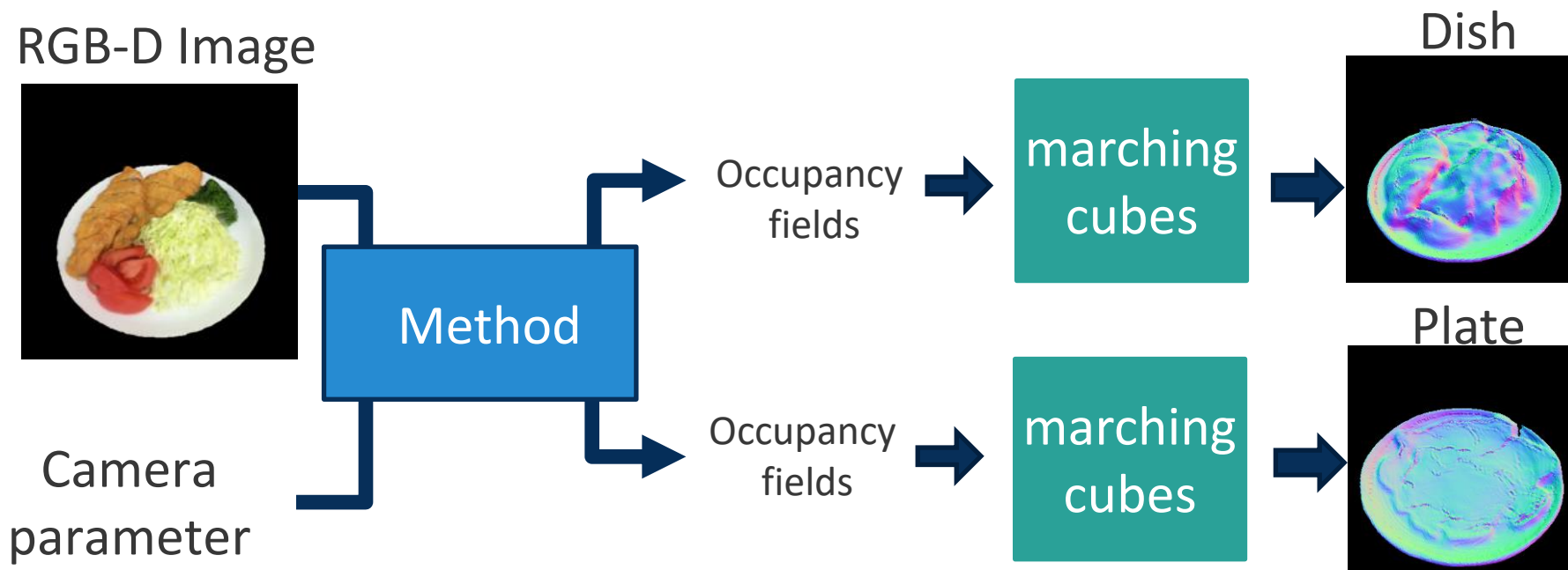


Hou らの手法[5]から引用

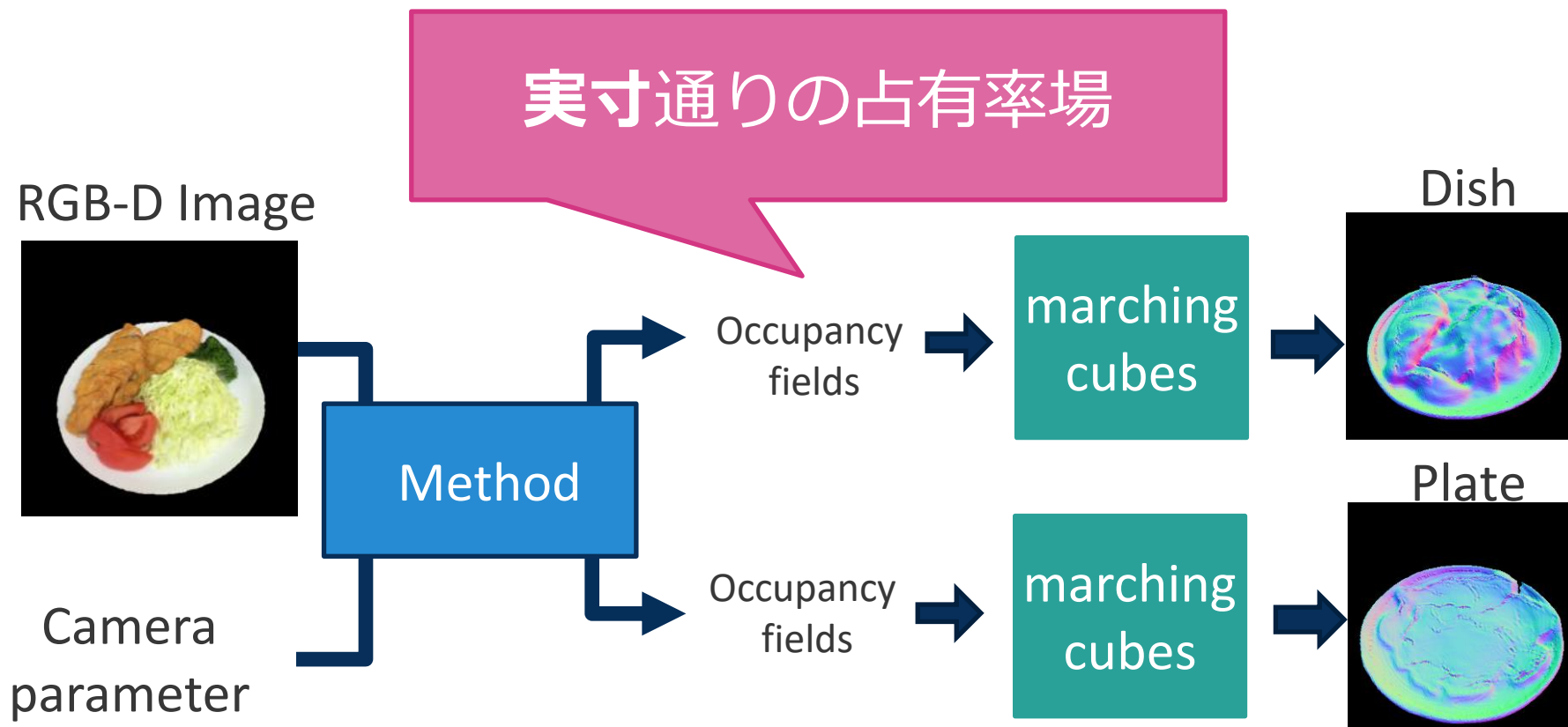
[4] RGBD Based Dimensional Decomposition Residual Network for 3D Semantic Scene Completion, CVPR 2019

[5] RevealNet: Seeing Behind Objects in RGB-D Scans, CVPR 2020

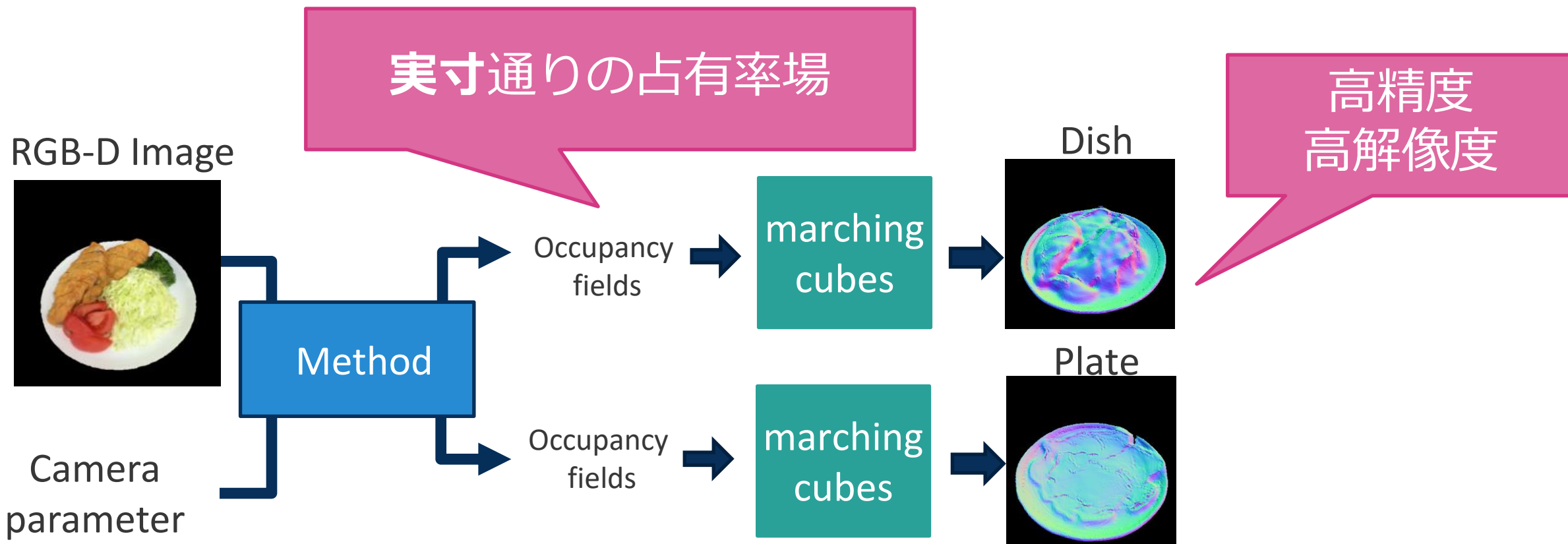
- **RGB-D**画像を用いて、**実寸通り**の食事と食器の三次元形状を**高精度/高解像度**に再構成する**陰関数表現**を用いた手法を提案。



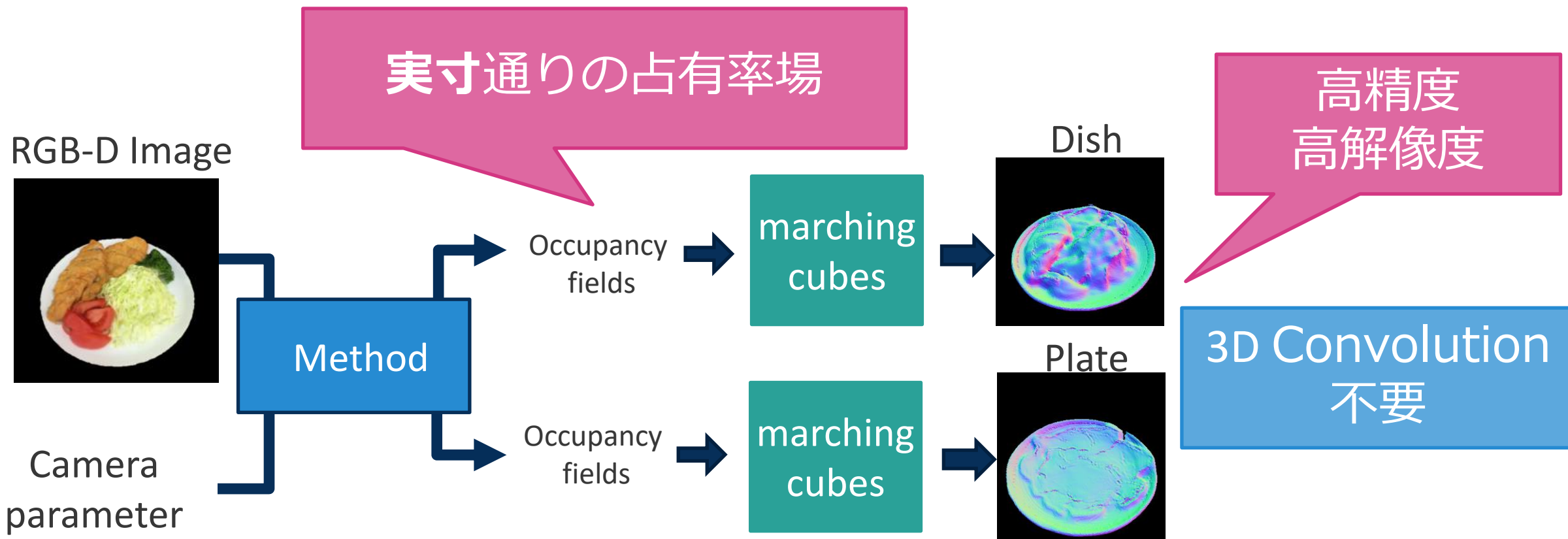
- **RGB-D**画像を用いて、**実寸通り**の食事と食器の三次元形状を**高精度/高解像度**に再構成する**陰関数表現**を用いた手法を提案。



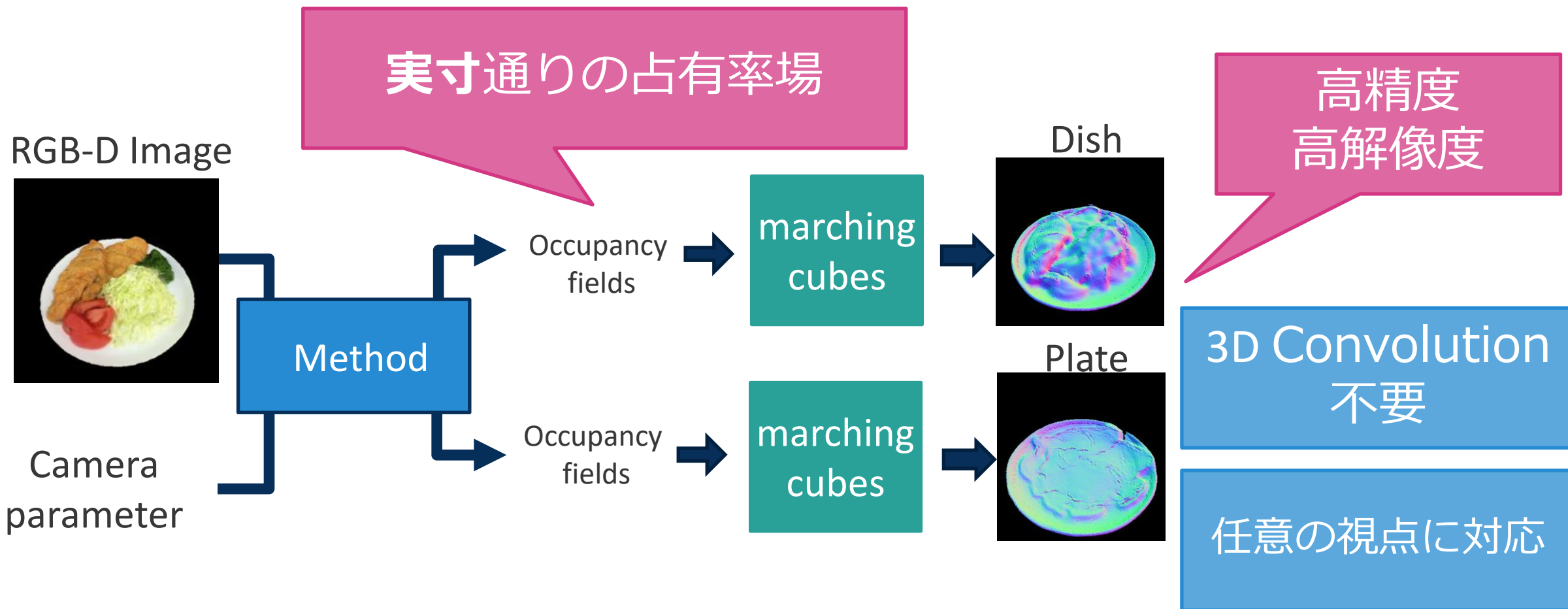
- **RGB-D**画像を用いて、**実寸通り**の食事と食器の三次元形状を**高精度/高解像度**に再構成する**陰関数表現**を用いた手法を提案。



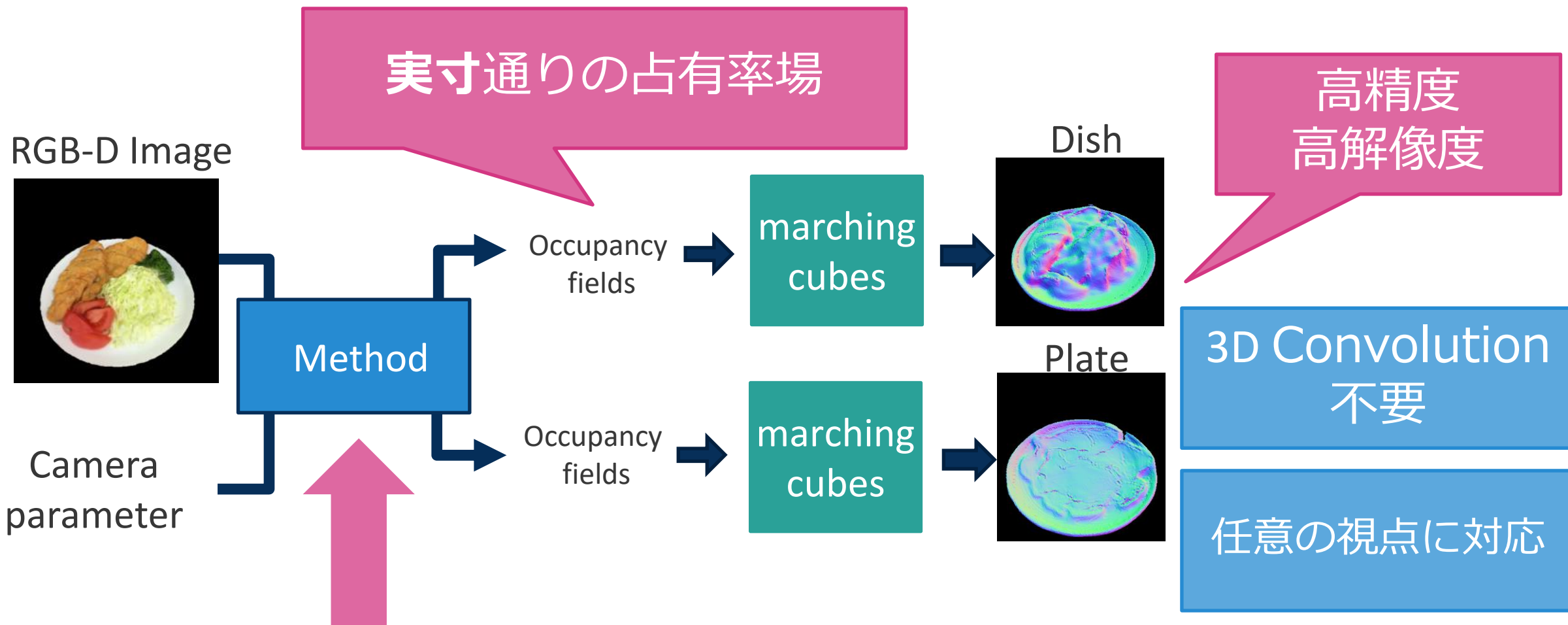
- **RGB-D**画像を用いて、**実寸通り**の食事と食器の三次元形状を**高精度/高解像度**に再構成する**陰関数表現**を用いた手法を提案。



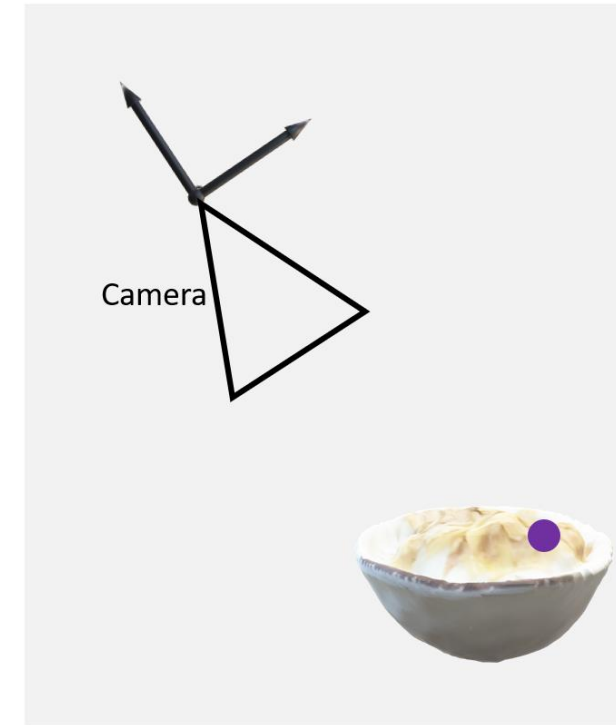
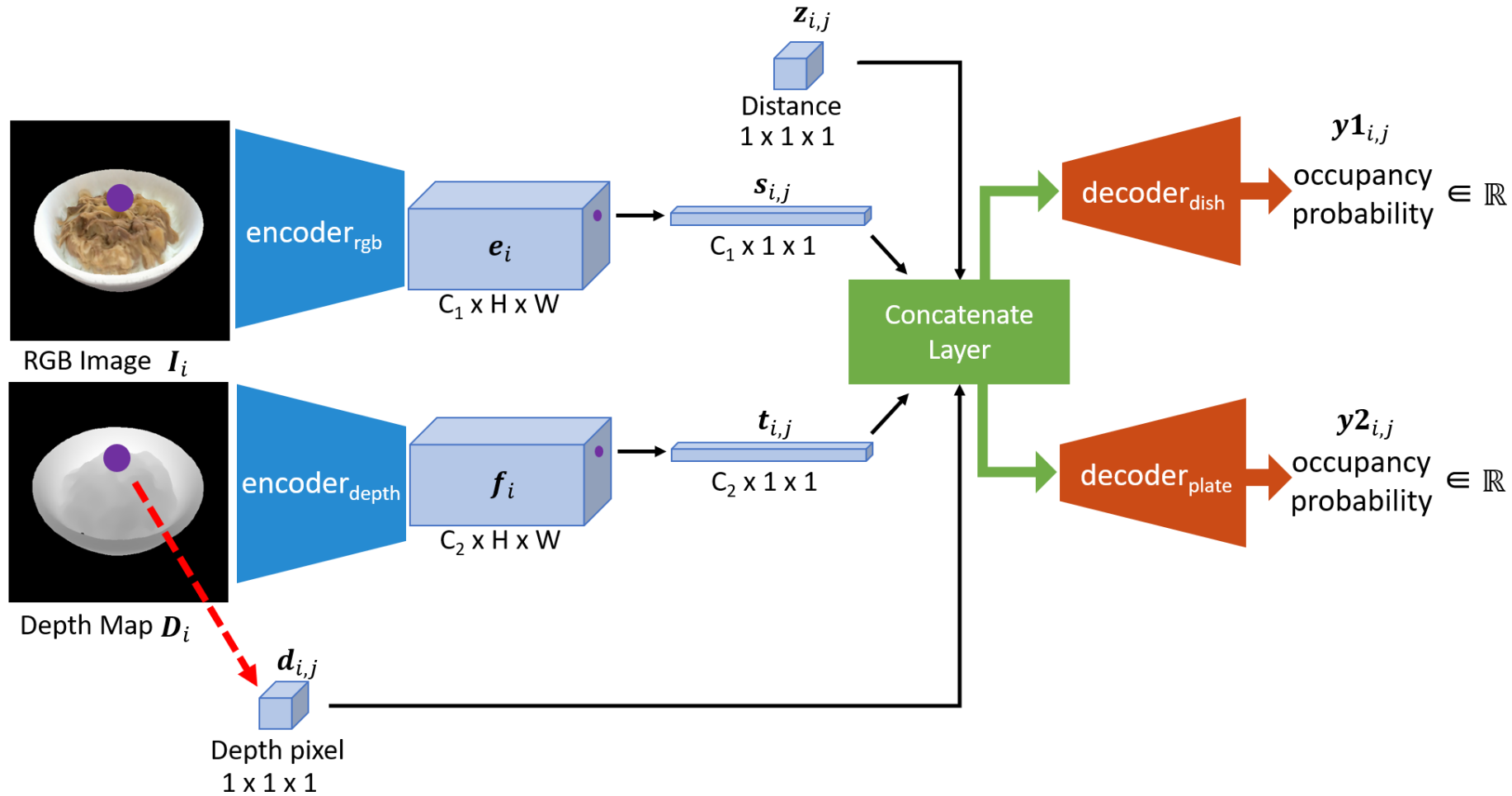
- **RGB-D**画像を用いて、**実寸通り**の食事と食器の三次元形状を**高精度/高解像度**に再構成する**陰関数表現**を用いた手法を提案。



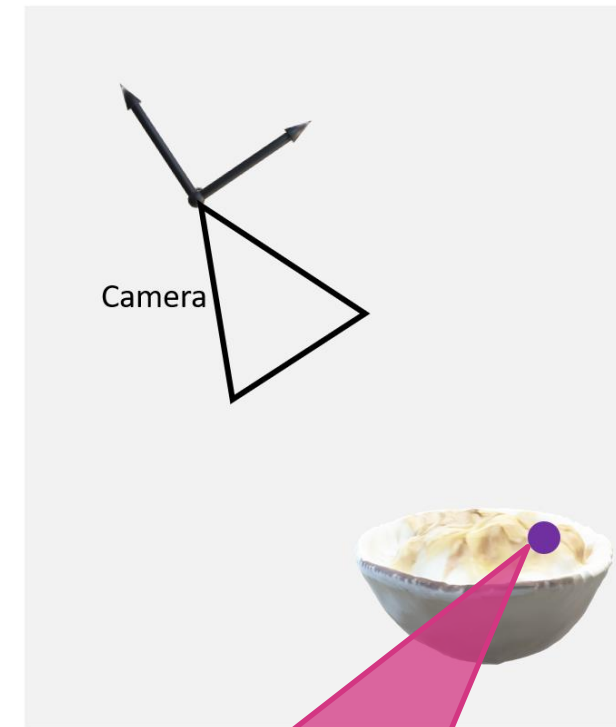
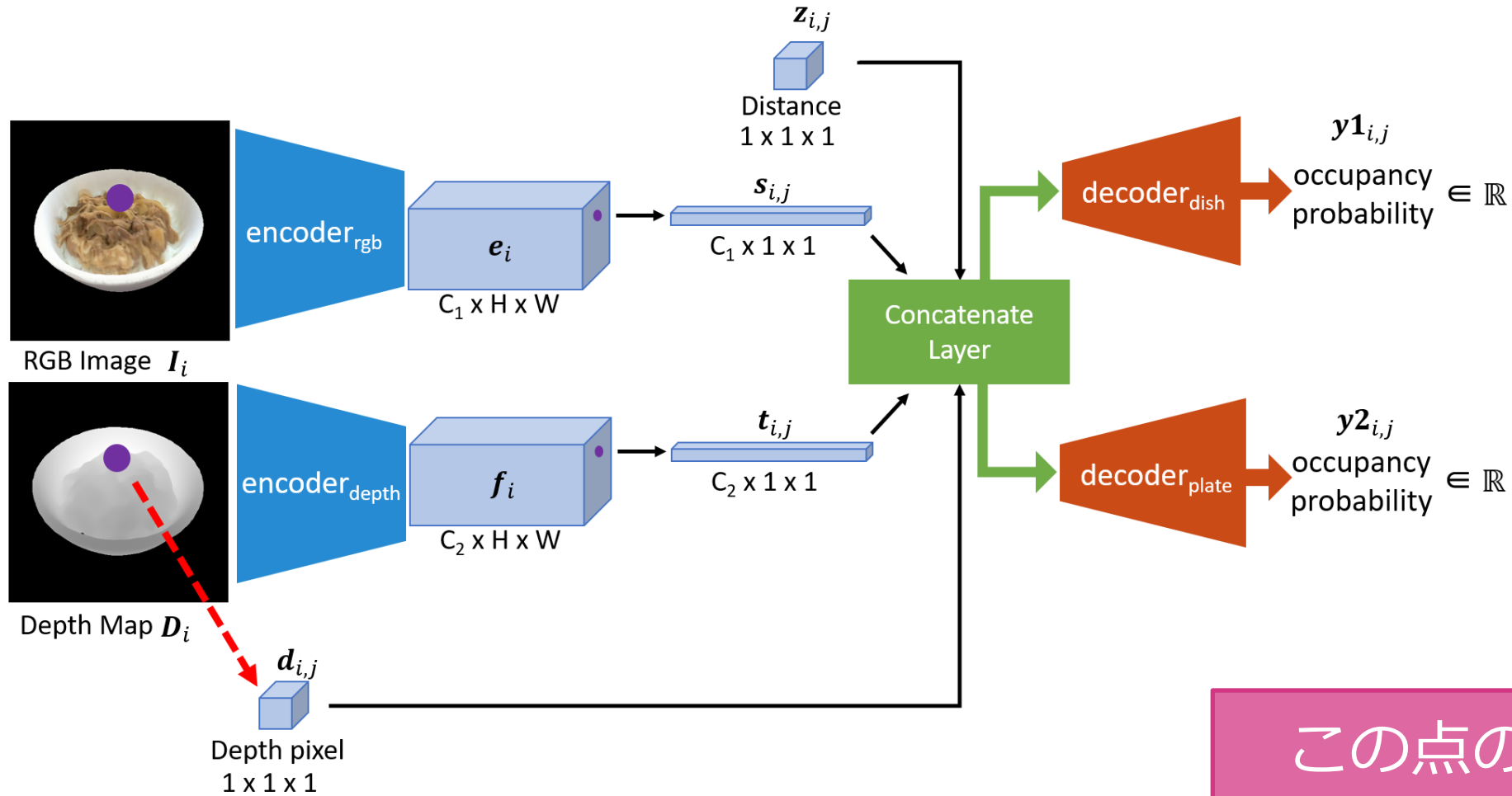
- **RGB-D**画像を用いて、**実寸通り**の食事と食器の三次元形状を**高精度/高解像度**に再構成する**陰関数表現**を用いた手法を提案。



ネットワーク概要図



ネットワーク概要図



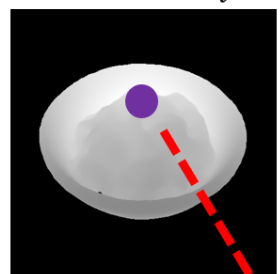
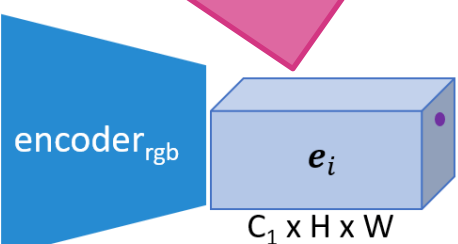
この点の占有率を推論

ネットワーク概要図

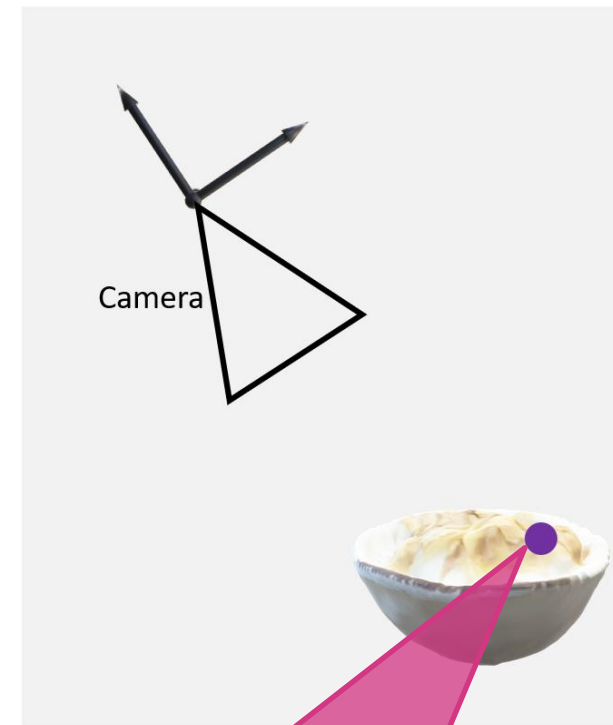
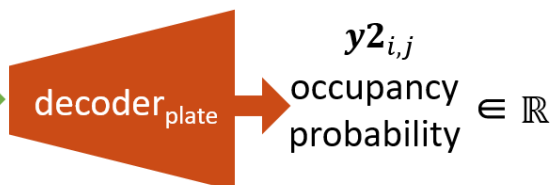
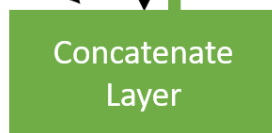
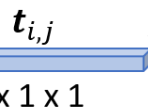
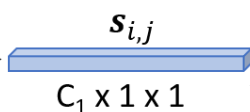
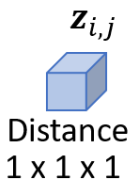
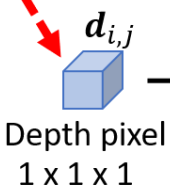
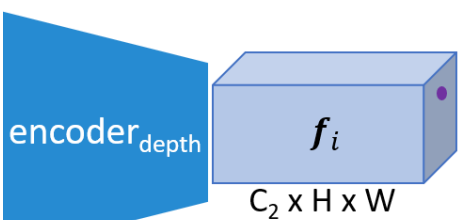
RGB-Dそれぞれの
特徴量を抽出



RGB Image I_i



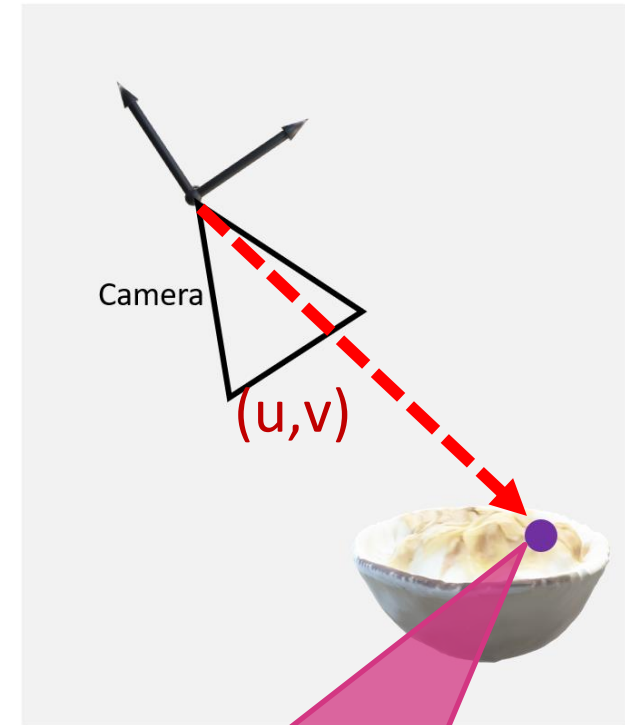
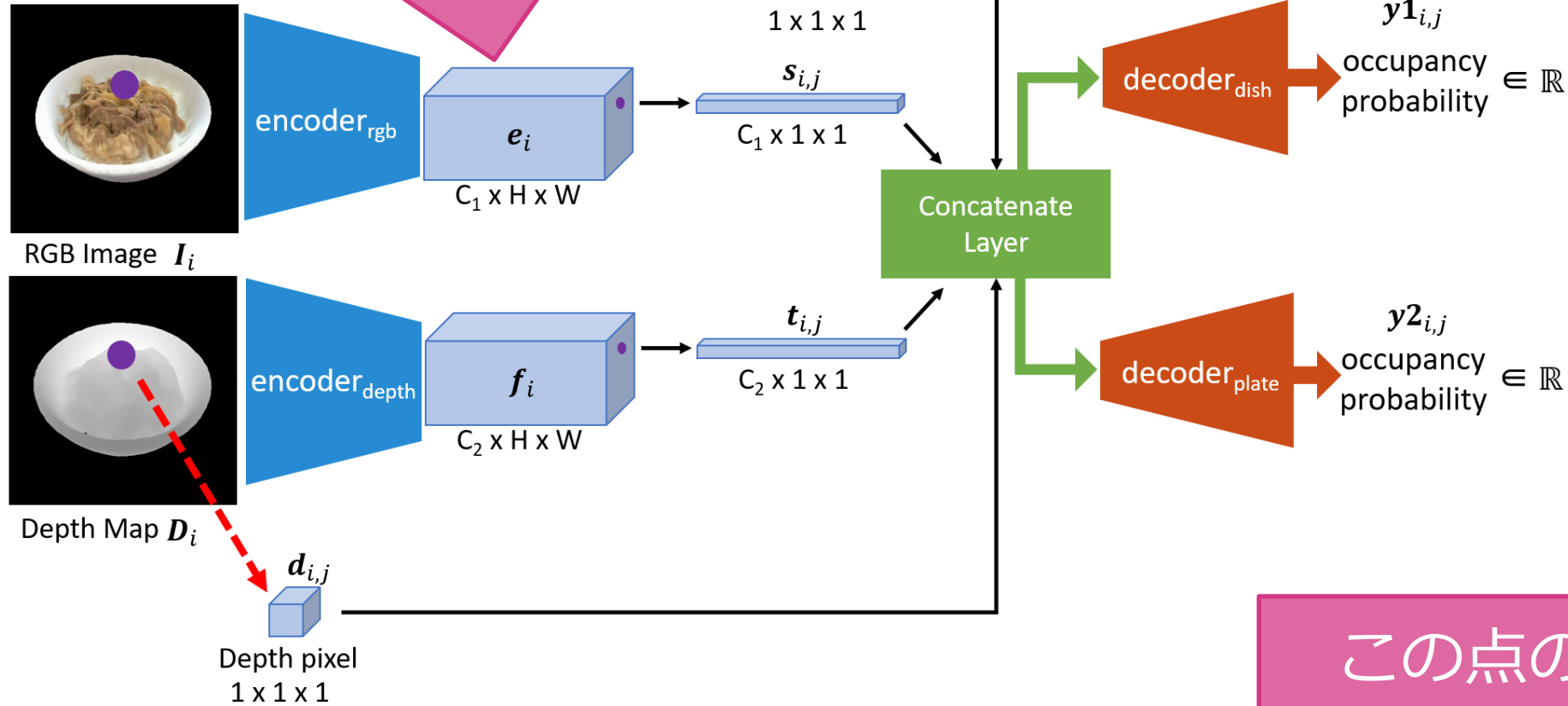
Depth Map D_i



この点の占有率を推論

ネットワーク概要図

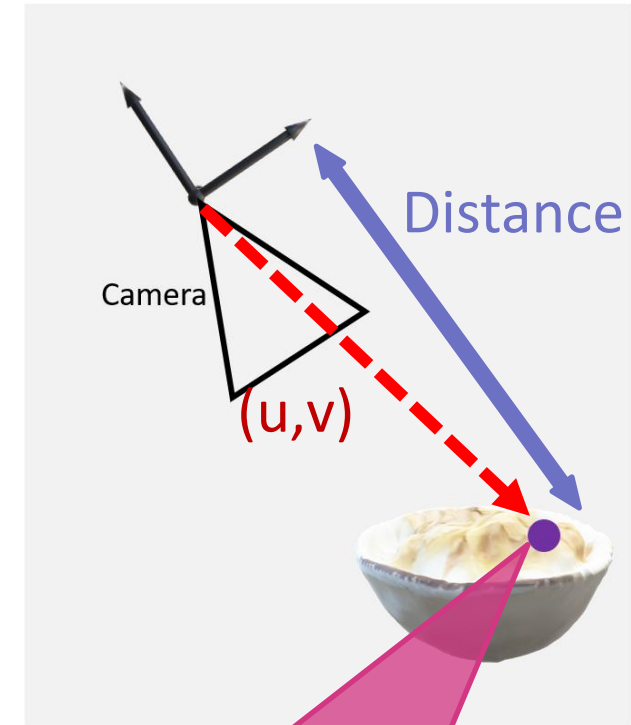
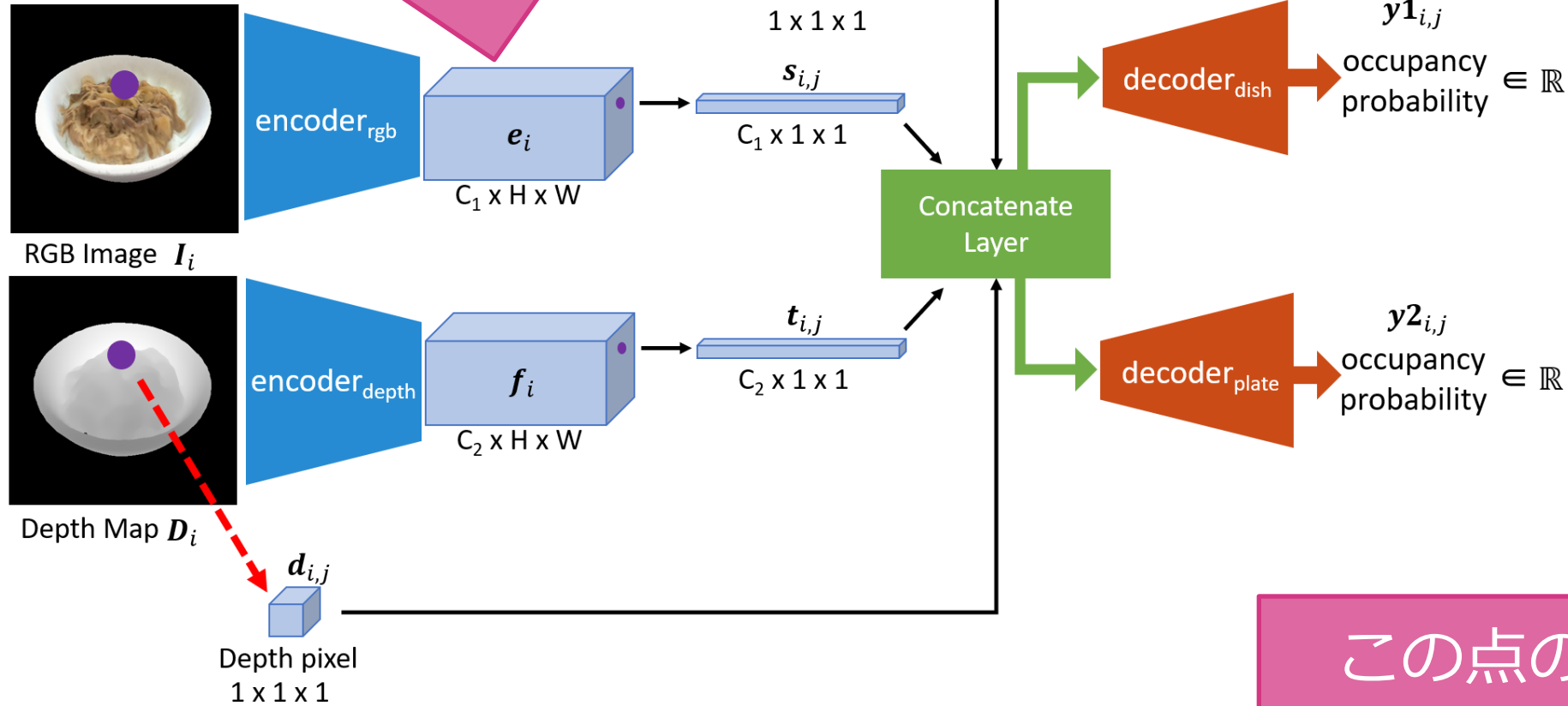
RGB-Dそれぞれの
特徴量を抽出



この点の占有率を推論

ネットワーク概要図

RGB-Dそれぞれの
特徴量を抽出

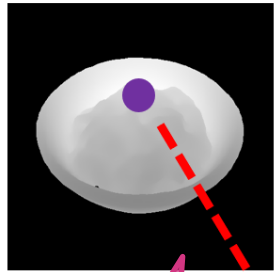
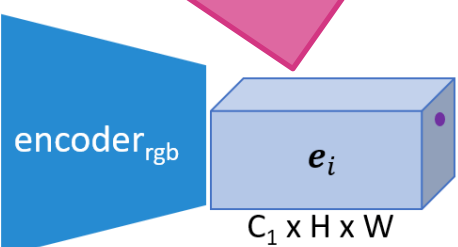


この点の占有率を推論

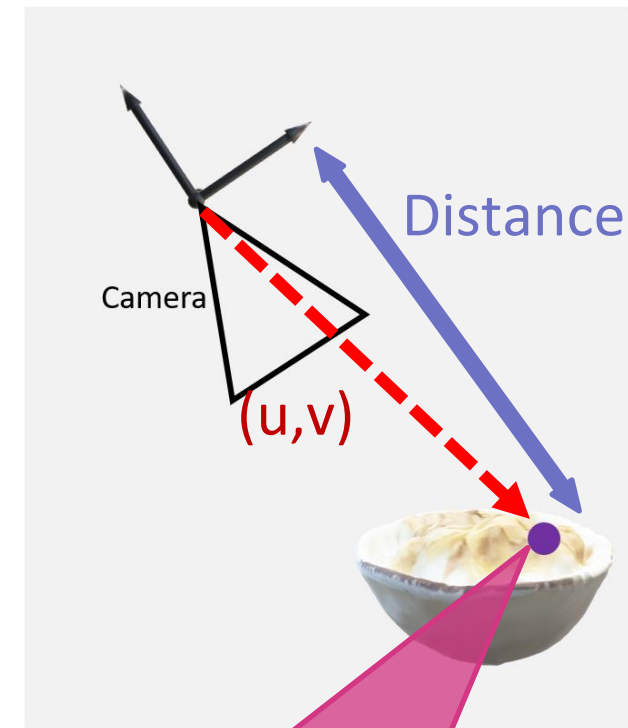
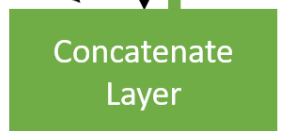
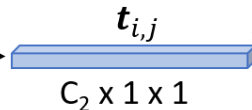
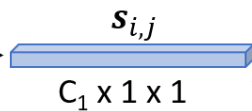
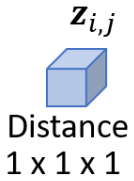
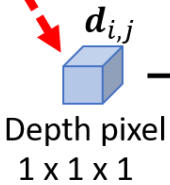
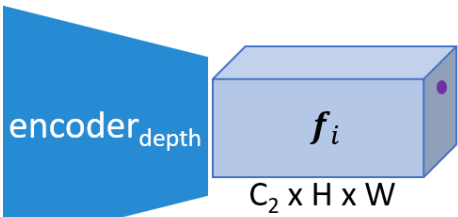
RGB-Dそれぞれの
特徴量を抽出



RGB Image I_i



Depth Map D_i



この点の占有率を推論

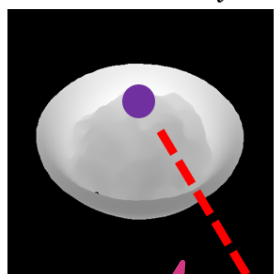
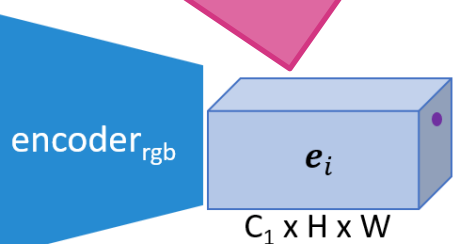
サンプリング

ネットワーク概要図

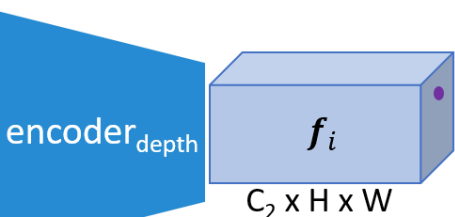
RGB-Dそれぞれの
特徴量を抽出



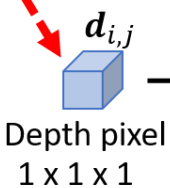
RGB Image I_i



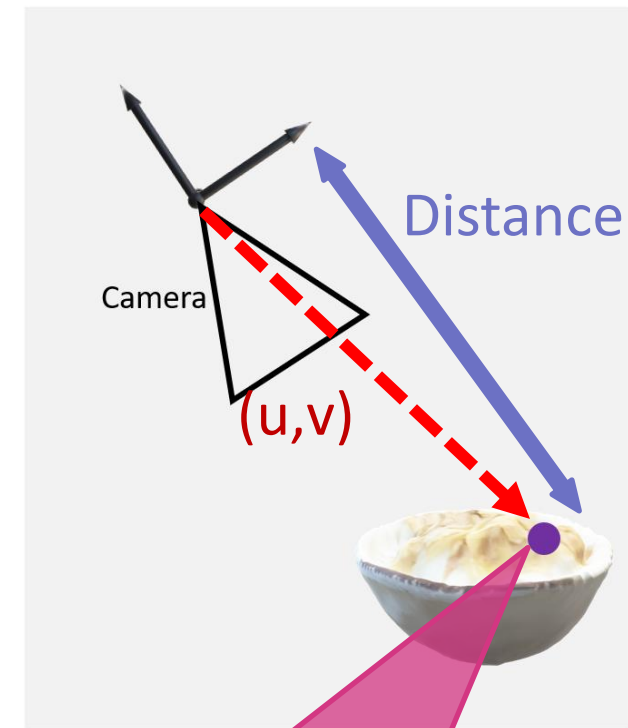
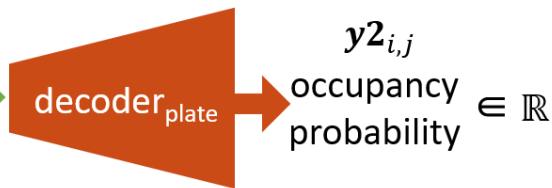
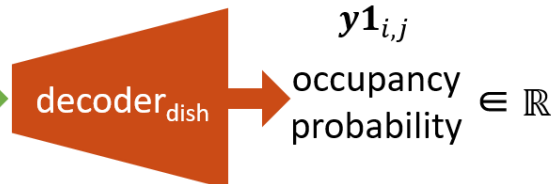
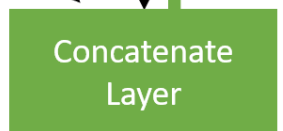
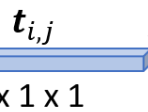
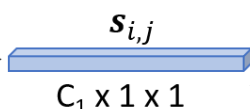
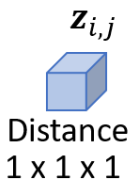
Depth Map D_i



サンプリング



サンプリング



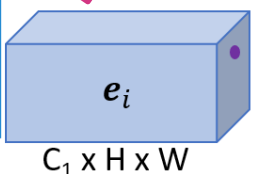
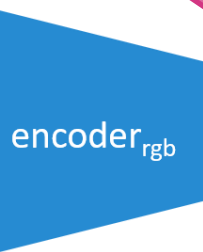
この点の占有率を推論

ネットワーク概要図

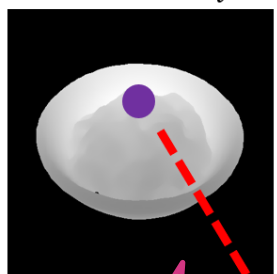
RGB-Dそれぞれの
特徴量を抽出



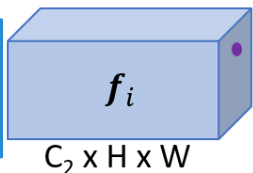
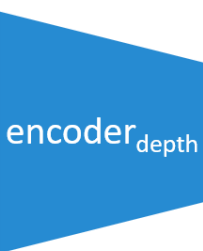
RGB Image I_i



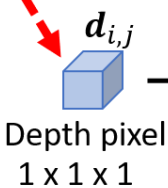
$C_1 \times H \times W$



Depth Map D_i



$C_2 \times H \times W$

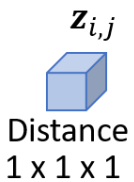


Depth pixel
 $1 \times 1 \times 1$

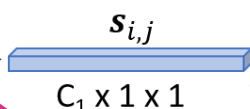
サンプリング

サンプリング

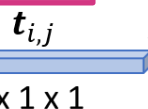
サンプリング



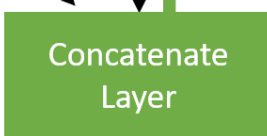
Distance
 $1 \times 1 \times 1$



$C_1 \times 1 \times 1$



$C_2 \times 1 \times 1$

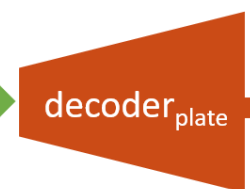


Concatenate
Layer



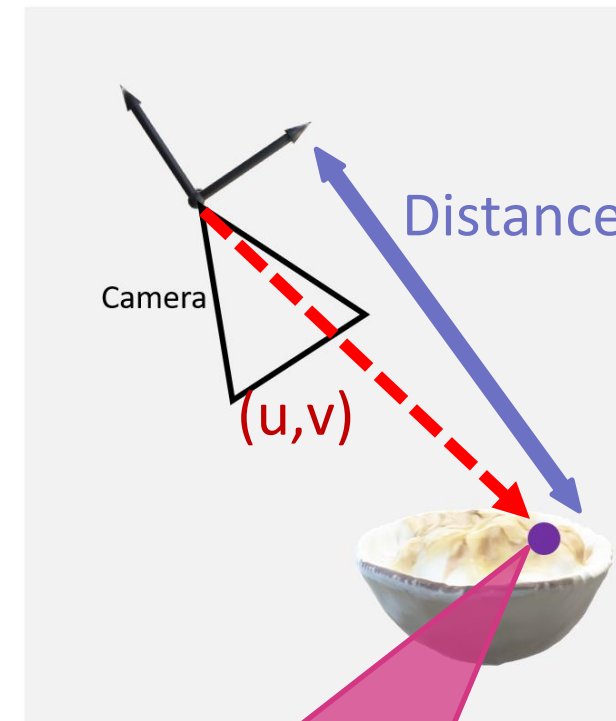
decoder_{dish}

$y1_{i,j}$
occupancy
probability $\in \mathbb{R}$



decoder_{plate}

$y2_{i,j}$
occupancy
probability $\in \mathbb{R}$



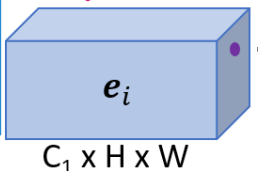
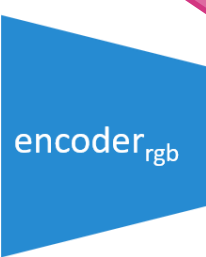
この点の占有率を推論

ネットワーク概要図

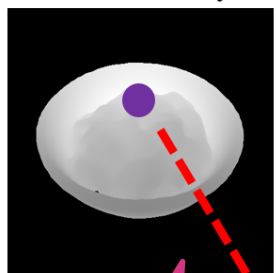
RGB-Dそれぞれの
特徴量を抽出



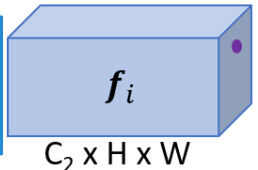
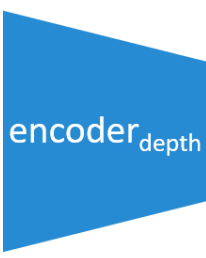
RGB Image I_i



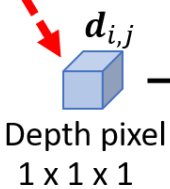
$C_1 \times H \times W$



Depth Map D_i



$C_2 \times H \times W$



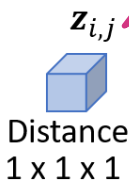
Depth pixel
 $1 \times 1 \times 1$

サンプリング

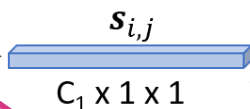
サンプリング

サンプリング

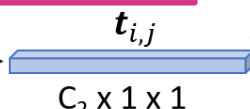
事前に計算した
distance



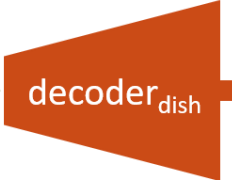
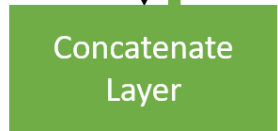
Distance
 $1 \times 1 \times 1$



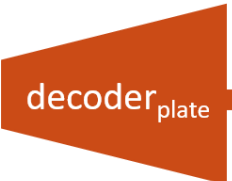
$C_1 \times 1 \times 1$



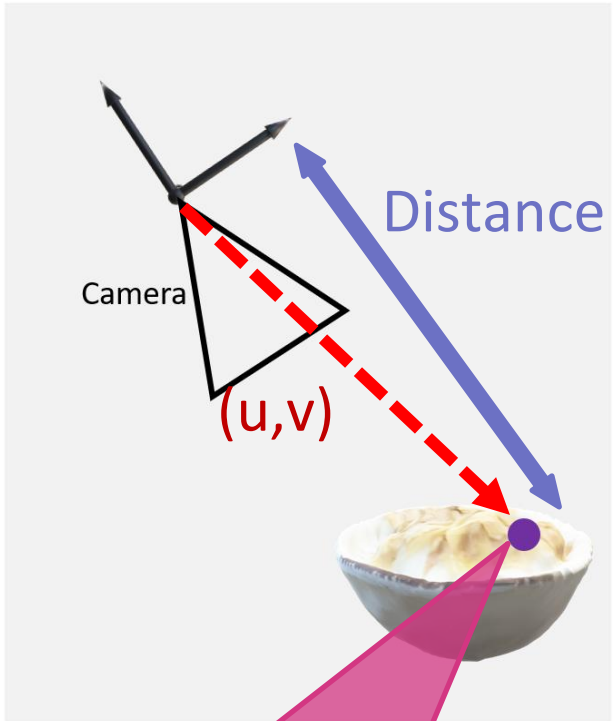
$C_2 \times 1 \times 1$



$y1_{i,j}$
occupancy
probability $\in \mathbb{R}$



$y2_{i,j}$
occupancy
probability $\in \mathbb{R}$



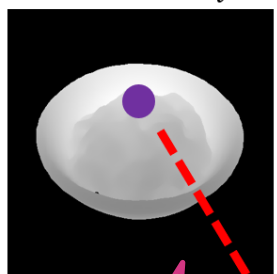
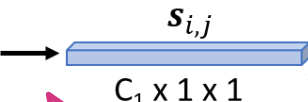
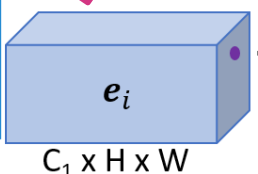
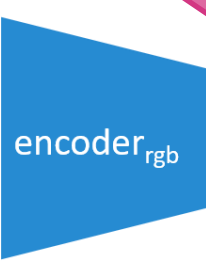
この点の占有率を推論

ネットワーク概要図

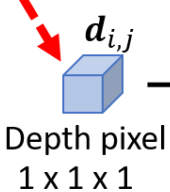
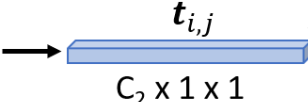
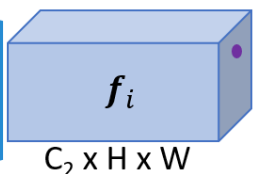
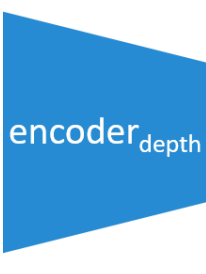
RGB-Dそれぞれの
特徴量を抽出



RGB Image I_i



Depth Map D_i



Depth pixel
 $1 \times 1 \times 1$

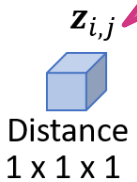
サンプリング

サンプリング

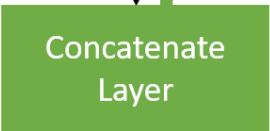
サンプリング

事前に計算した
distance

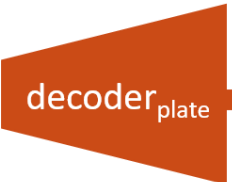
占有率を推論



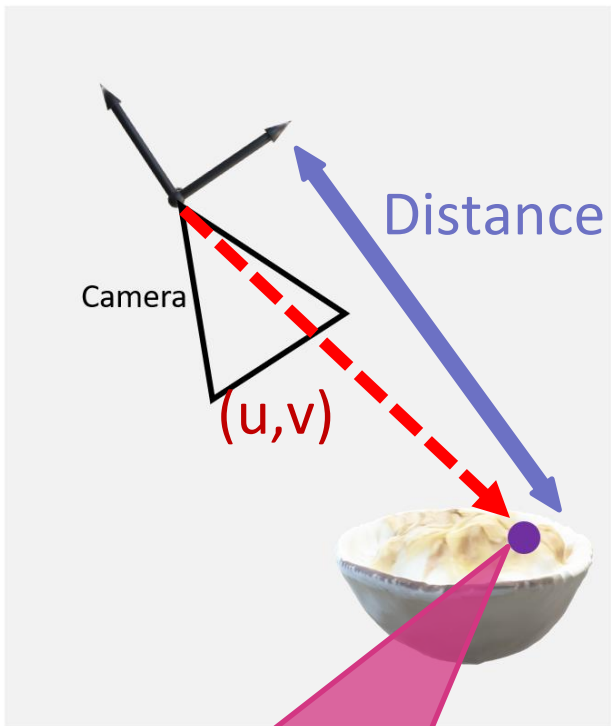
Distance
 $1 \times 1 \times 1$



$y1_{i,j}$
occupancy
probability $\in \mathbb{R}$



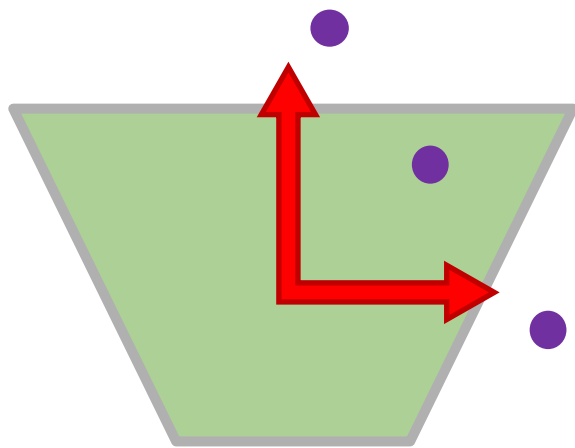
$y2_{i,j}$
occupancy
probability $\in \mathbb{R}$



この点の占有率を推論

**実寸通りの再構成を実現するために
三次元形状の正規化を行わない事で発生する課題**

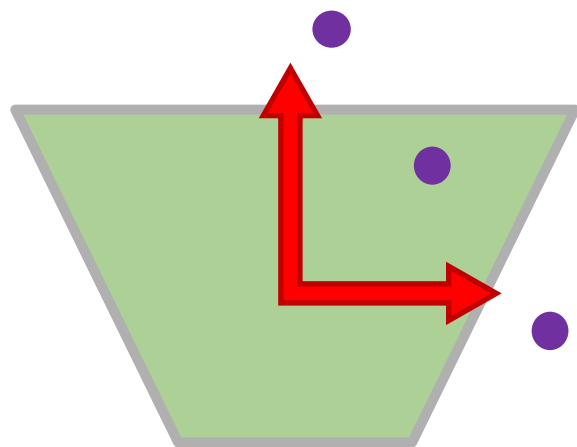
- 本来**正規化**を行う事で学習を可能にする
 - X, Y, Z 軸それぞれ一定の区間 (例えば $-0.5 \sim 0.5$) の空間で占有率場を推論。
 - 物体に含まれていそうな座標を学習できる。



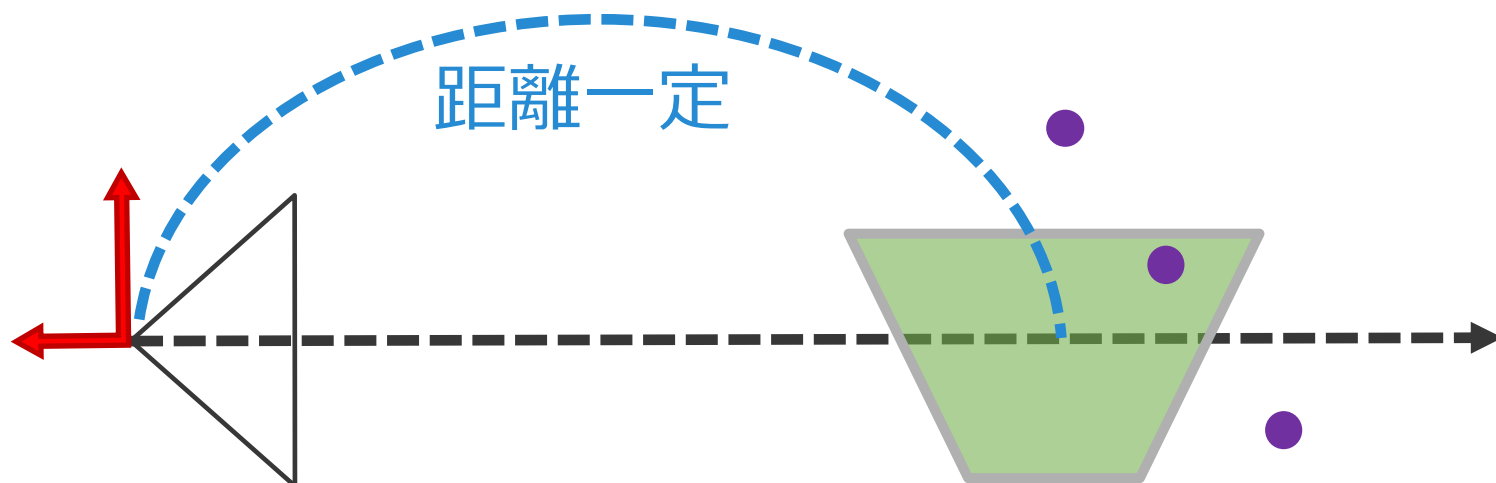
- ①物体の大きさ
- ②原点が物体中心を正規化

深度画像の活用

- 本来**正規化**を行う事で学習を可能にする
 - X, Y, Z 軸それぞれ一定の区間 (例えば $-0.5 \sim 0.5$) の空間で占有率場を推論。
 - 物体に含まれていそうな座標を学習できる。



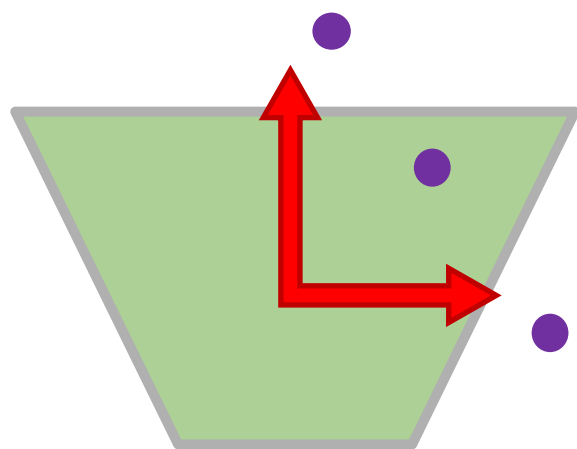
- ①物体の大きさ
- ②原点が物体中心を正規化



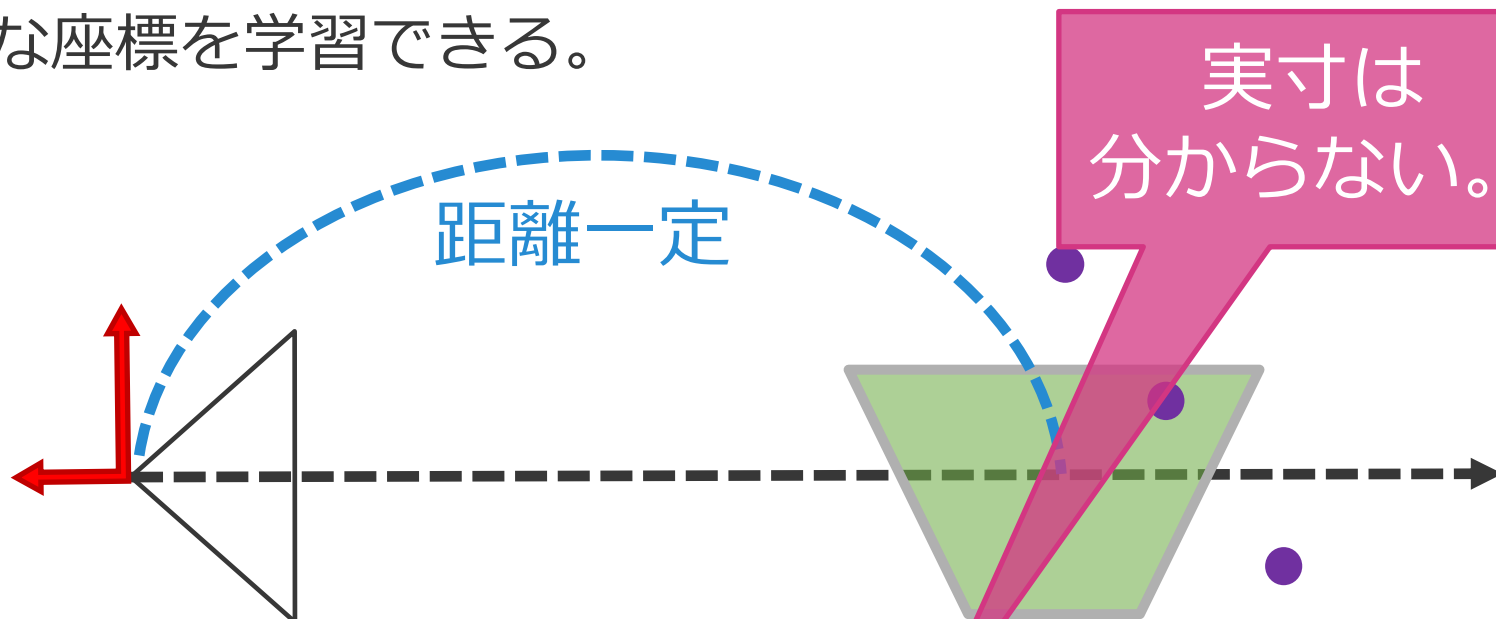
- ①物体の大きさ、
- ②カメラと物体までの距離を正規化(PIFu[3])

深度画像の活用

- 本来**正規化**を行う事で学習を可能にする
 - X, Y, Z 軸それぞれ一定の区間 (例えば $-0.5 \sim 0.5$) の空間で占有率場を推論。
 - 物体に含まれていそうな座標を学習できる。

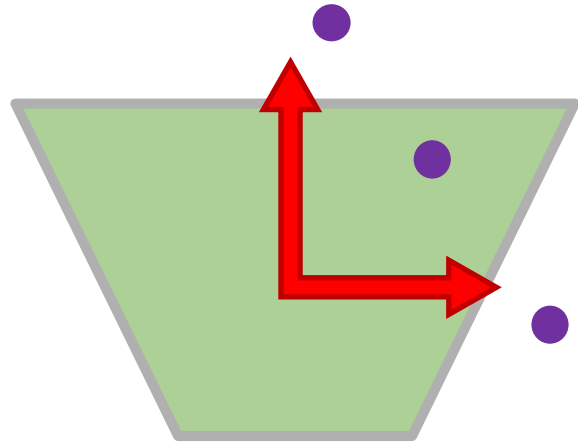
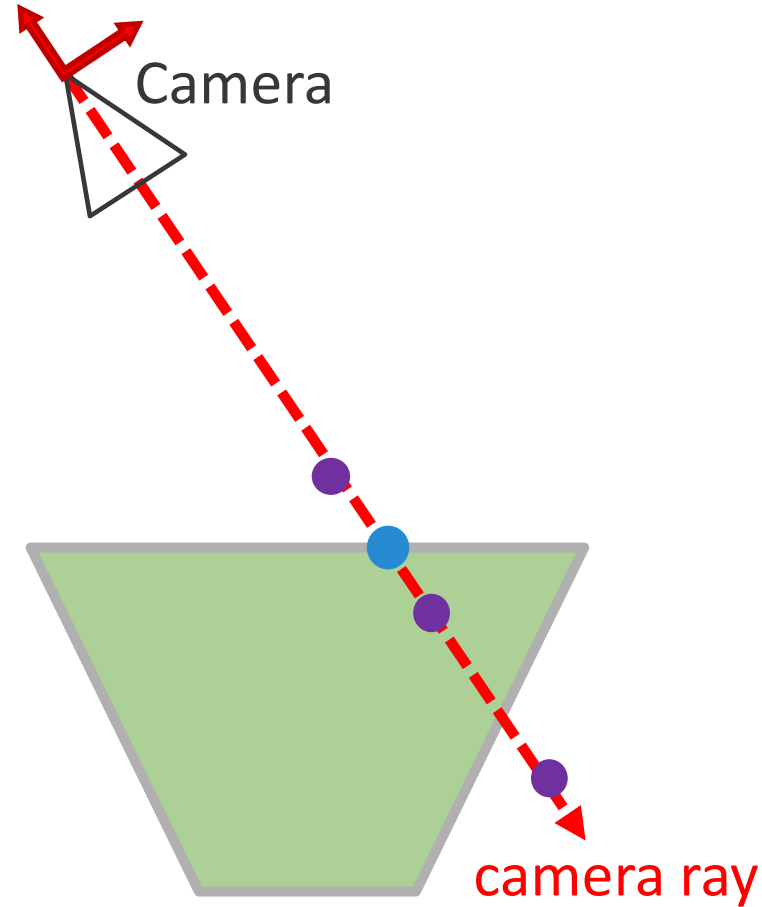


- ①物体の大きさ
- ②原点が物体中心を正規化



- ①物体の大きさ、
- ②カメラと物体までの距離を正規化(PIFu[3])

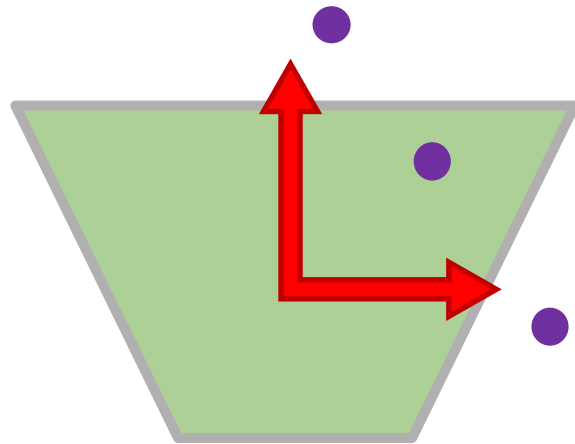
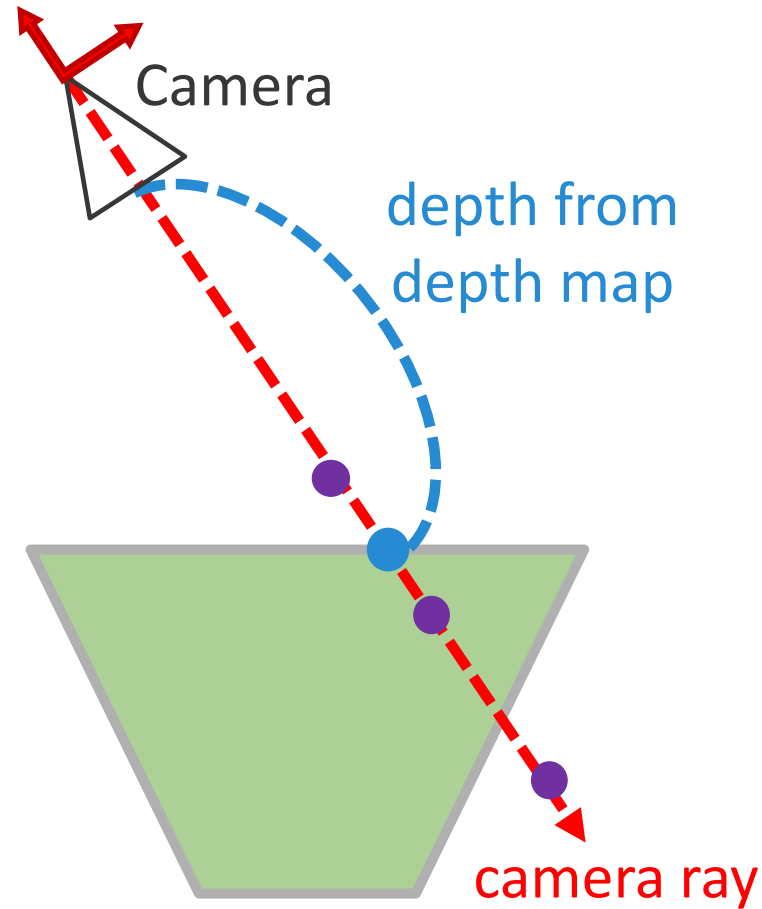
可視表面までの距離のサンプリング



正規化している場合
(絶対的)

本手法
(相対的)

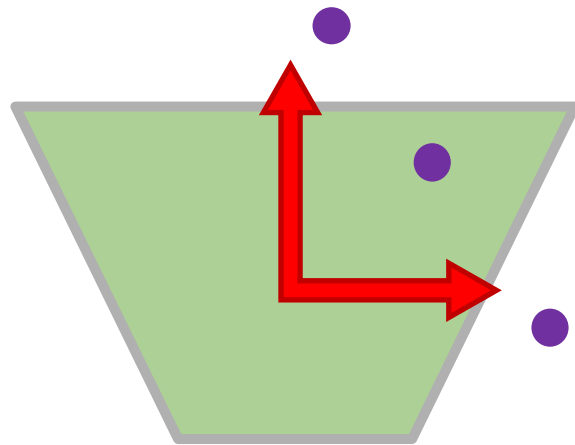
可視表面までの距離のサンプリング



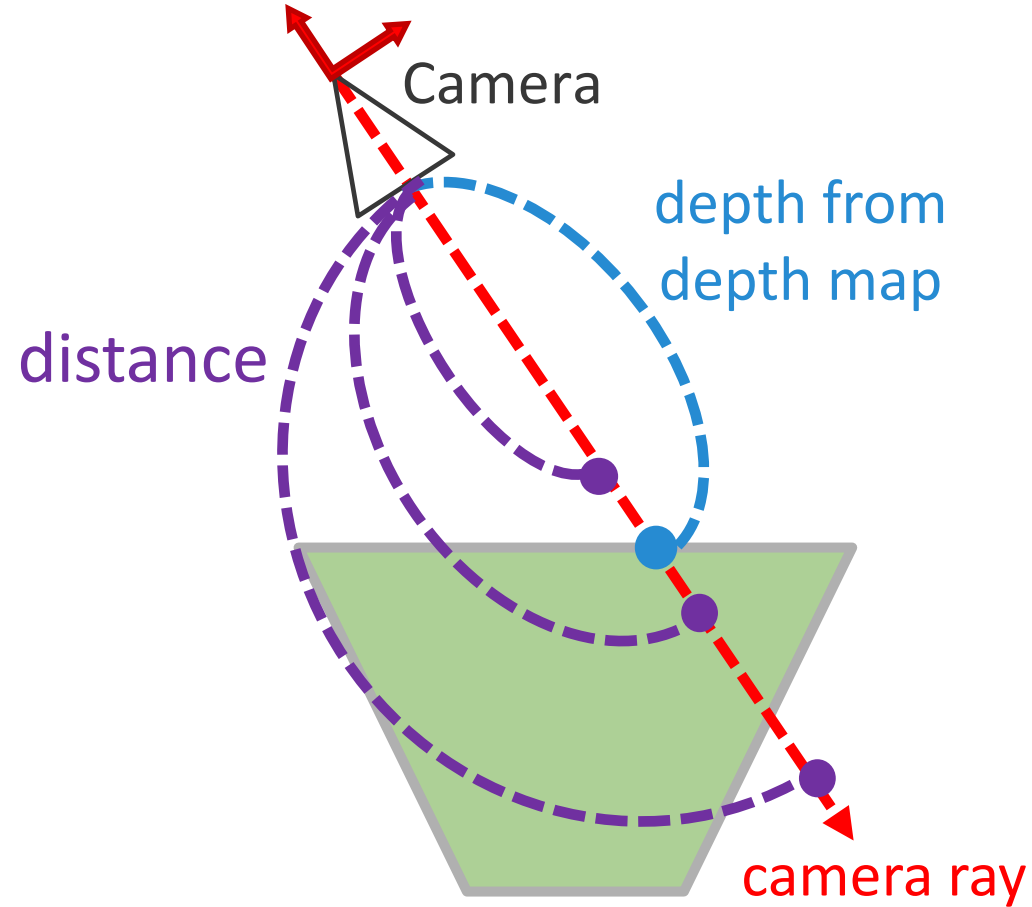
正規化している場合
(絶対的)

本手法
(相対的)

可視表面までの距離のサンプリング

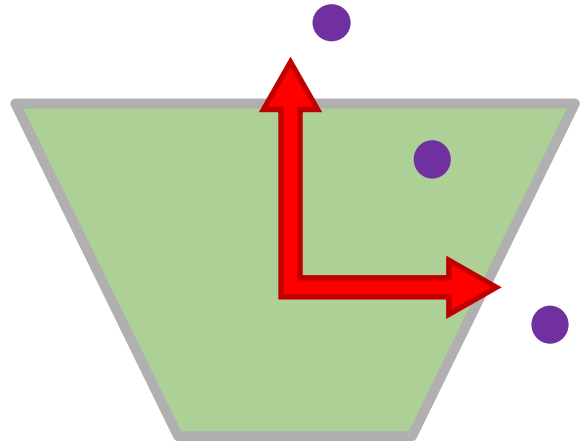


正規化している場合
(絶対的)

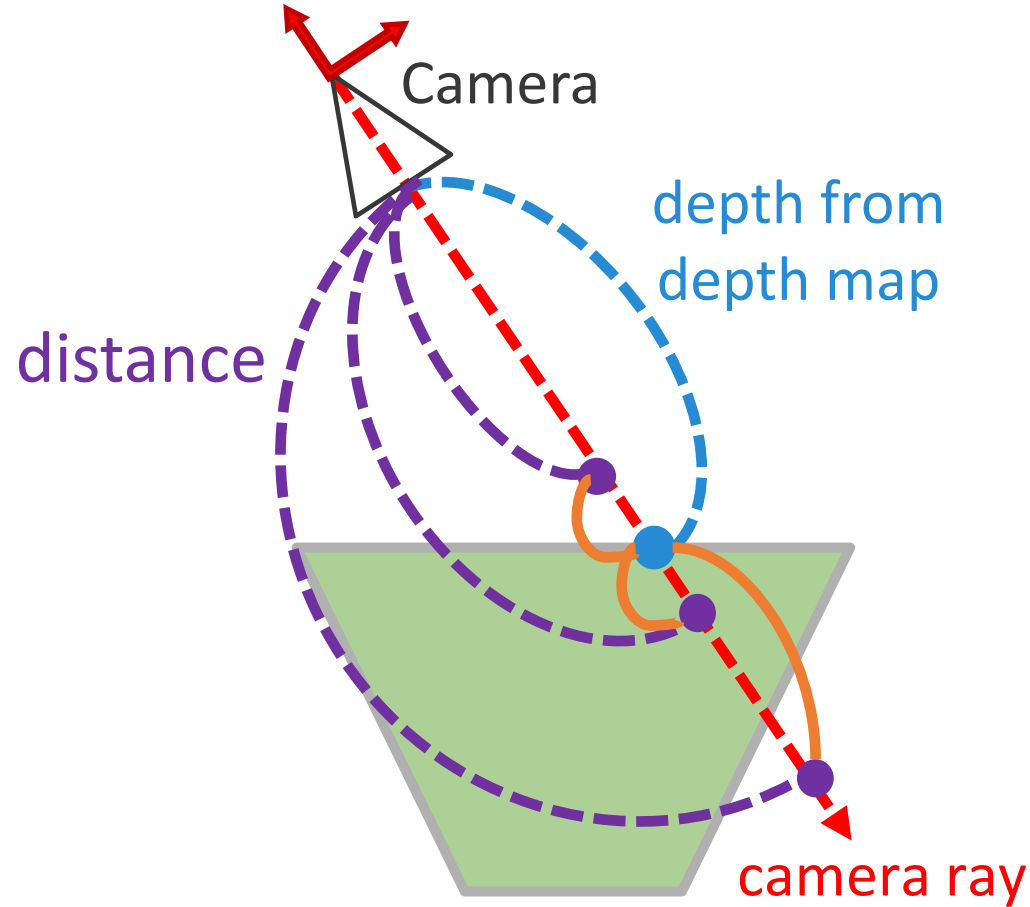


本手法
(相対的)

可視表面までの距離のサンプリング

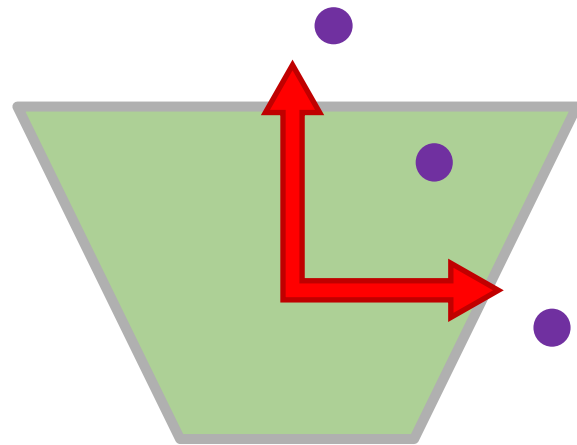


正規化している場合
(絶対的)

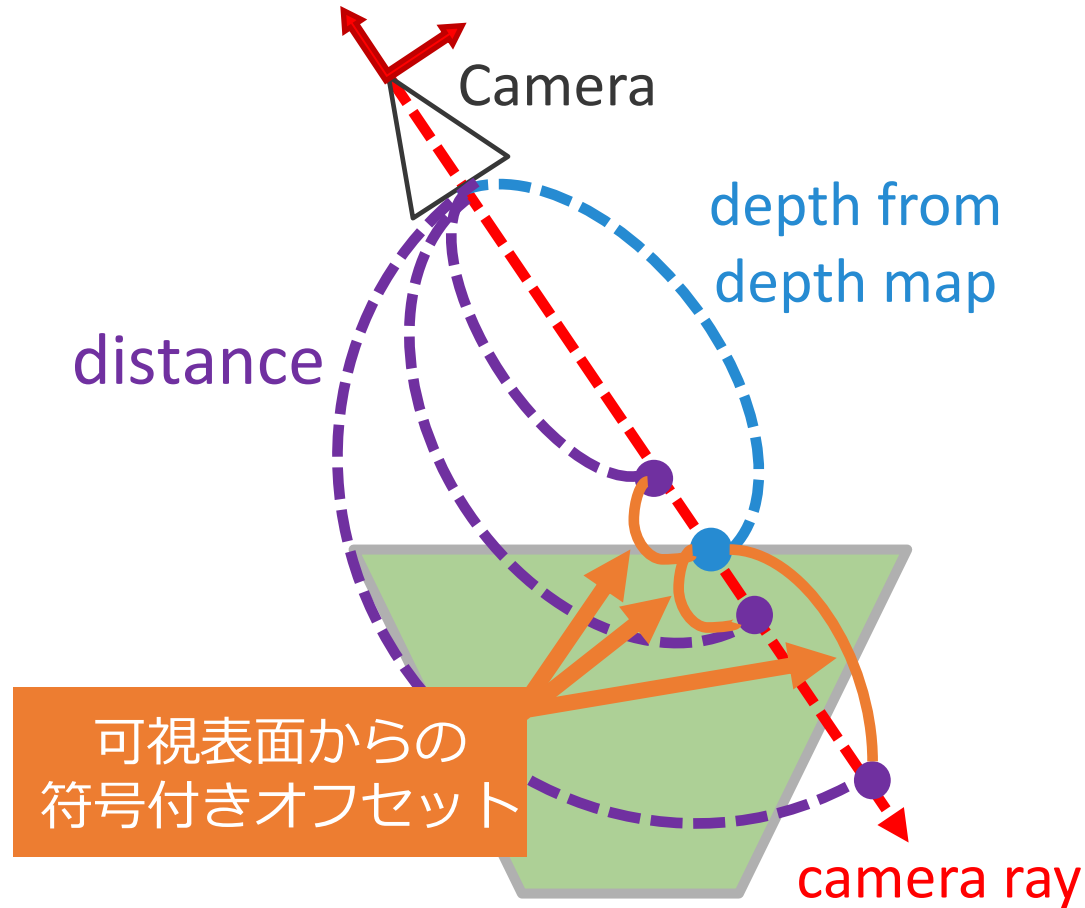


本手法
(相対的)

可視表面までの距離のサンプリング

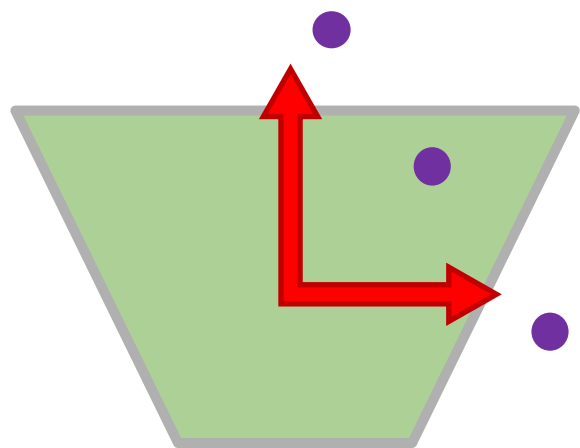


正規化している場合
(絶対的)

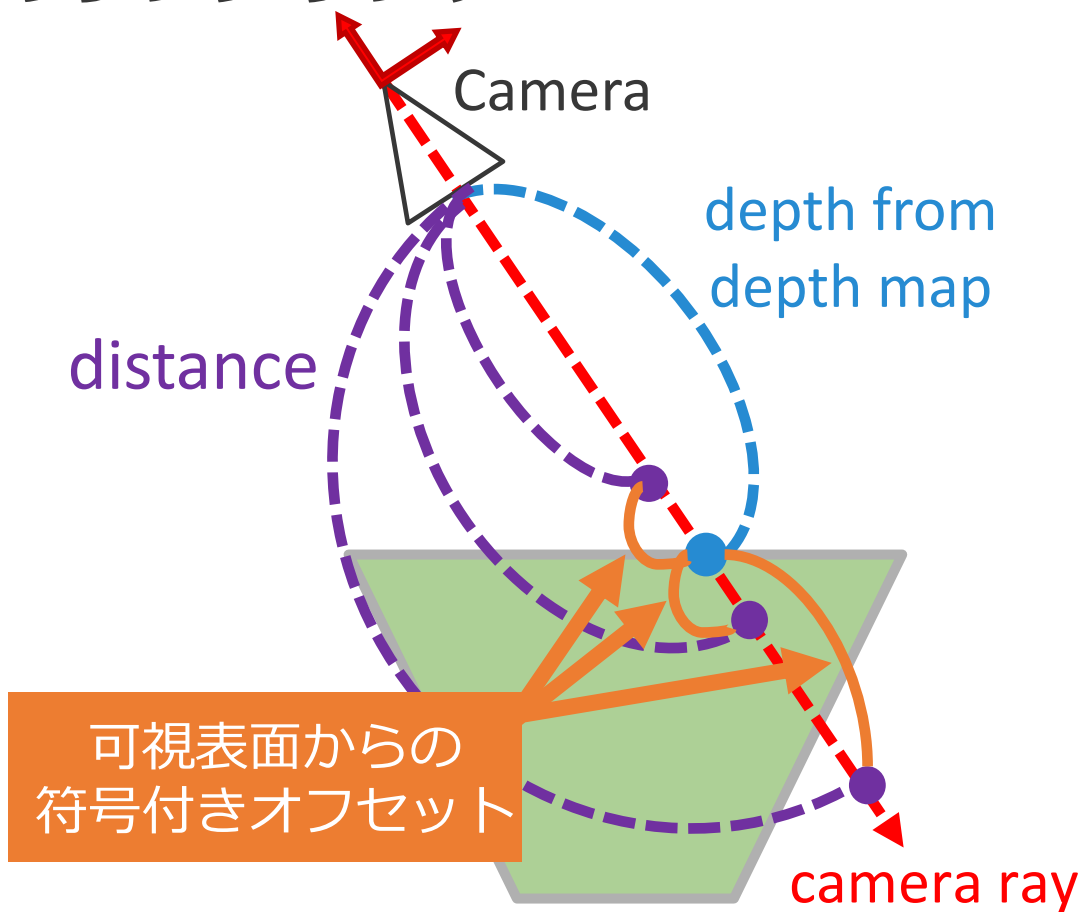


本手法
(相対的)

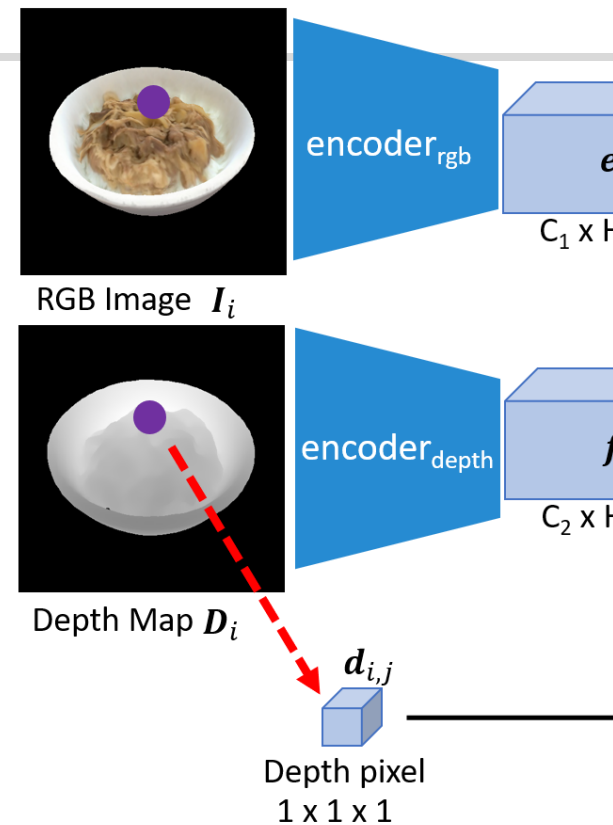
可視表面までの距離のサンプリング



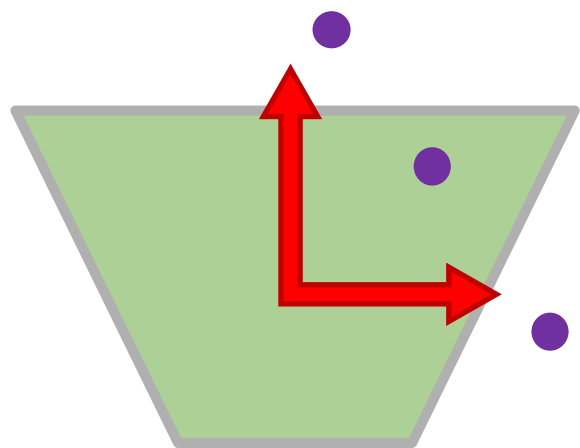
正規化している場合
(絶対的)



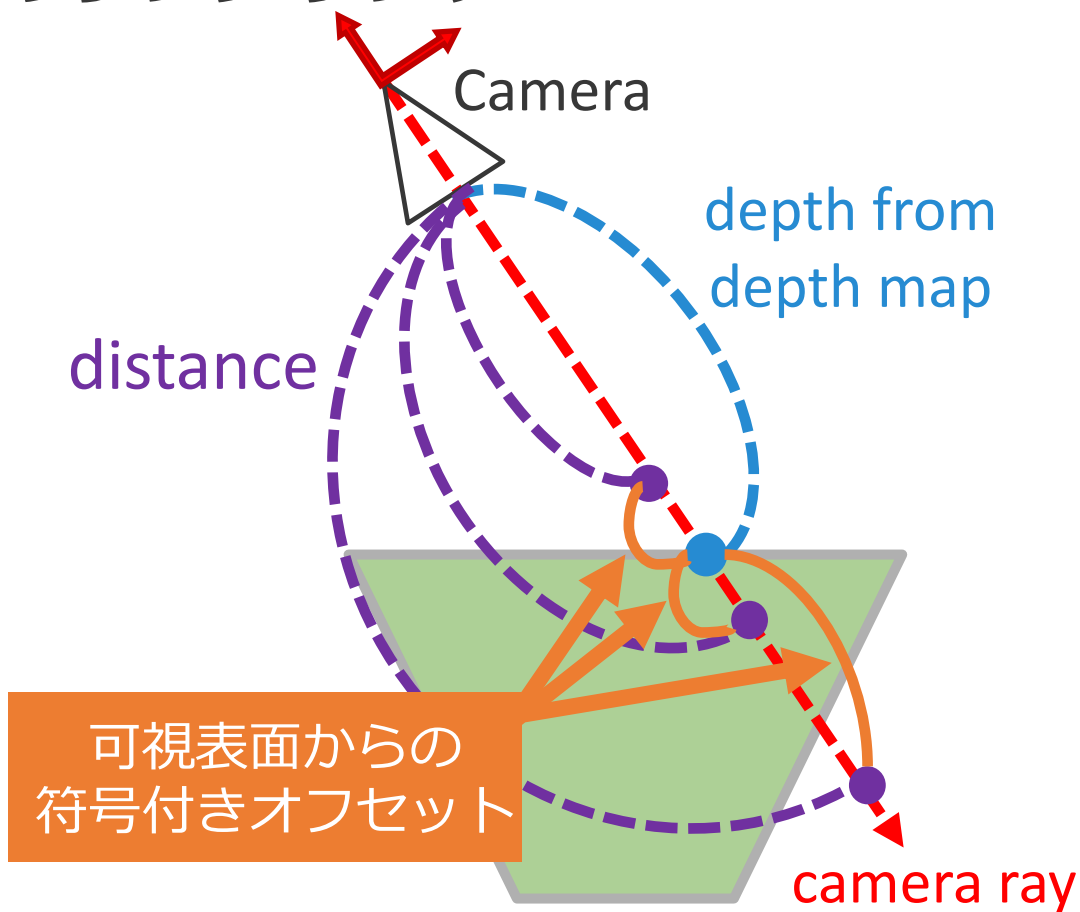
本手法
(相対的)



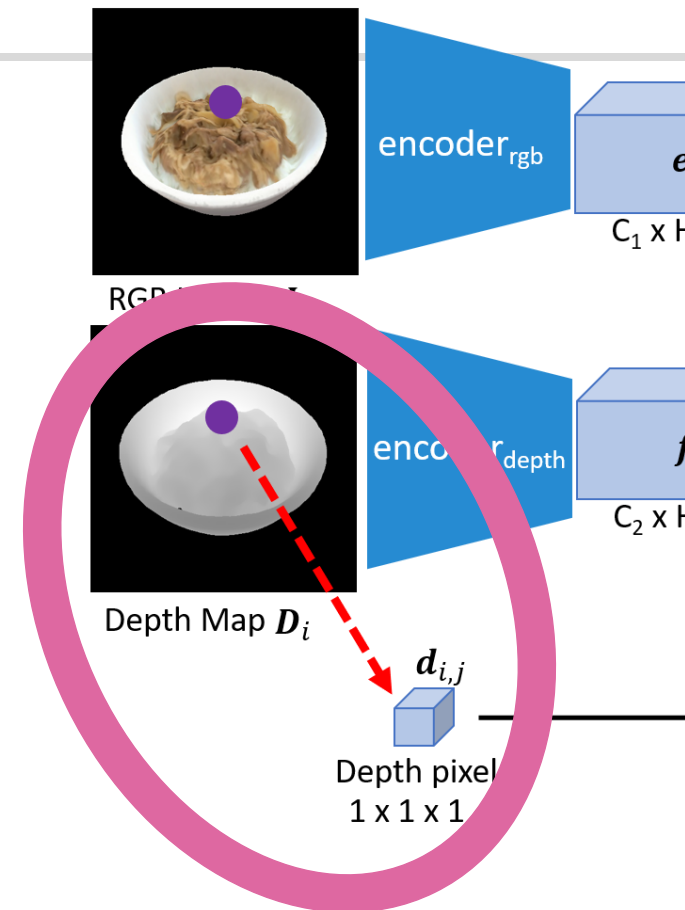
可視表面までの距離のサンプリング



正規化している場合
(絶対的)



本手法
(相対的)

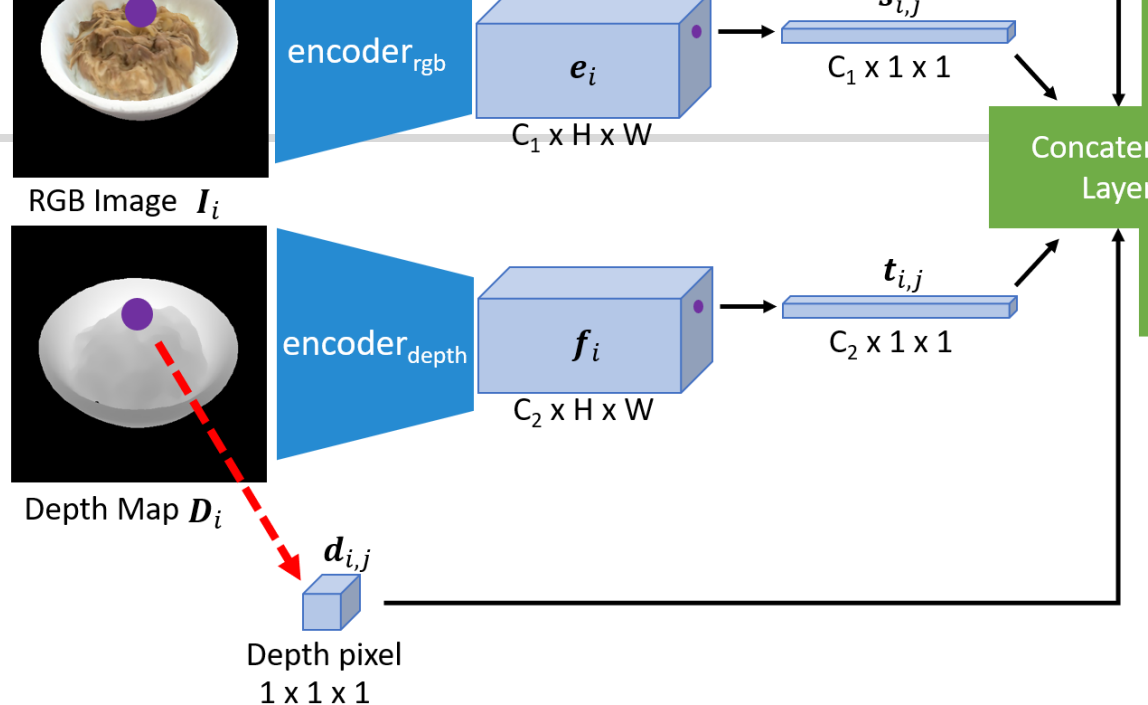


形状に関する特徴量の抽出

- 深度値サンプリング
 - ピクセル単位の情報
- ピクセル周辺の表面形状に関する特徴を見逃している。
- CNNで抽出して利用。

形状に関する特徴量の抽出

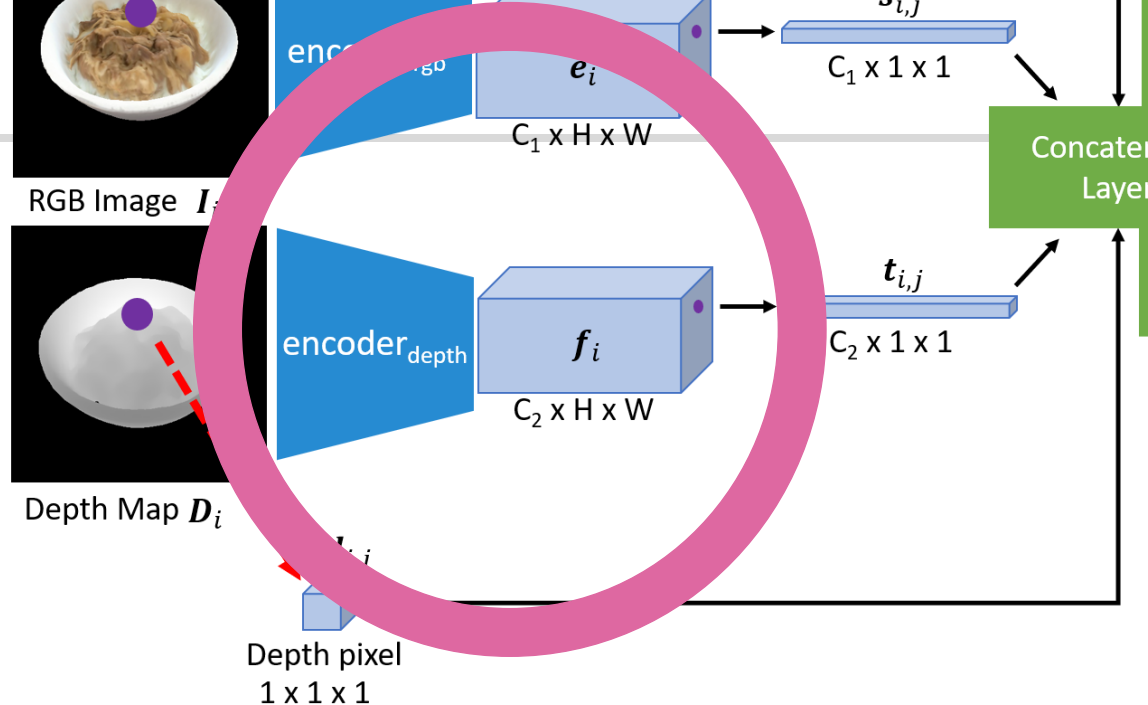
- 今までの活用方法
 - ピクセルレベルでサンプリング。



- ピクセル周辺の表面形状に関する特徴を見逃している。
- CNNで抽出して利用。

形状に関する特徴量の抽出

- 今までの活用方法
 - ピクセルレベルでサンプリング。



- ピクセル周辺の表面形状に関する特徴を見逃している。
- CNNで抽出して利用。

学習

$$\mathbf{e}_i = \text{encoder}_{rgb}(I_i) \quad (4.2)$$

$$\mathbf{f}_i = \text{encoder}_{depth}(D_i) \quad (4.3)$$

$$(u, v)_{i,j} = \text{projection}(p_{i,j}, K_i, R_i, T_i) \quad (4.4)$$

$$\mathbf{s}_{i,j} = \text{sample}(\mathbf{e}_i, (u, v)_{i,j}) \quad (4.5)$$

$$\mathbf{t}_{i,j} = \text{sample}(\mathbf{f}_i, (u, v)_{i,j}) \quad (4.6)$$

$$\mathbf{d}_{i,j} = \text{sample}(D_i, (u, v)_{i,j}) \quad (4.7)$$

$$\mathbf{z}_{i,j} = \text{distance}(p_{i,j}, K_i, R_i, T_i) \quad (4.8)$$

$$\mathbf{c}_{i,j} = \text{concatenate}(\mathbf{s}_{i,j}, \mathbf{t}_{i,j}, \mathbf{d}_{i,j}, \mathbf{z}_{i,j}) \quad (4.9)$$

$$y1_{i,j} = \text{decoder}_{dish}(\mathbf{c}_{i,j}) \quad (4.10)$$

$$y2_{i,j} = \text{decoder}_{plate}(\mathbf{c}_{i,j}) \quad (4.11)$$

最終的なロスは
Hungry Networksと同様

$$\mathcal{L}_O(\hat{o}, o) = \mathcal{L}_{bce}(\hat{o}, o) \quad (4.12)$$

$$\mathcal{L}_C(o1, o2) = \max(o2 - o1, 0) \quad (4.13)$$

$$\mathcal{L}_B = \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \sum_{j=1}^K \left(\lambda_1 \mathcal{L}_O(y1_{i,j}, o1_i(p_{i,j})) \right. \\ \left. + \lambda_2 \mathcal{L}_O(y2_{i,j}, o2_i(p_{i,j})) \right. \\ \left. + \lambda_3 \mathcal{L}_C(y1_{i,j}, y2_{i,j}) \right) \quad (4.14)$$

RGB-D 画像のレンダリング

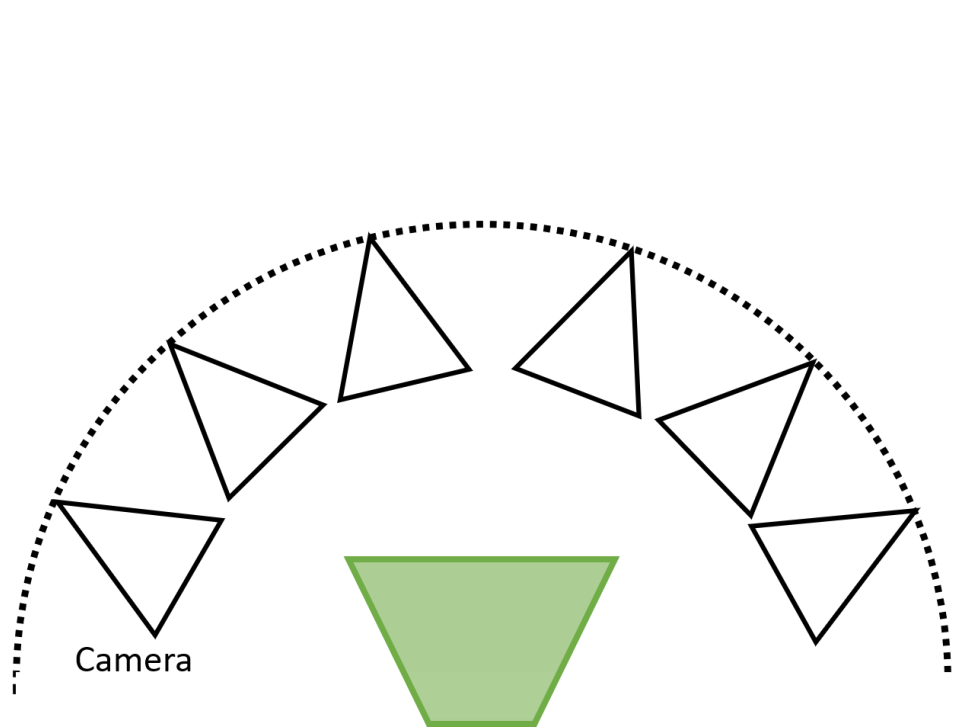


Image dataset A

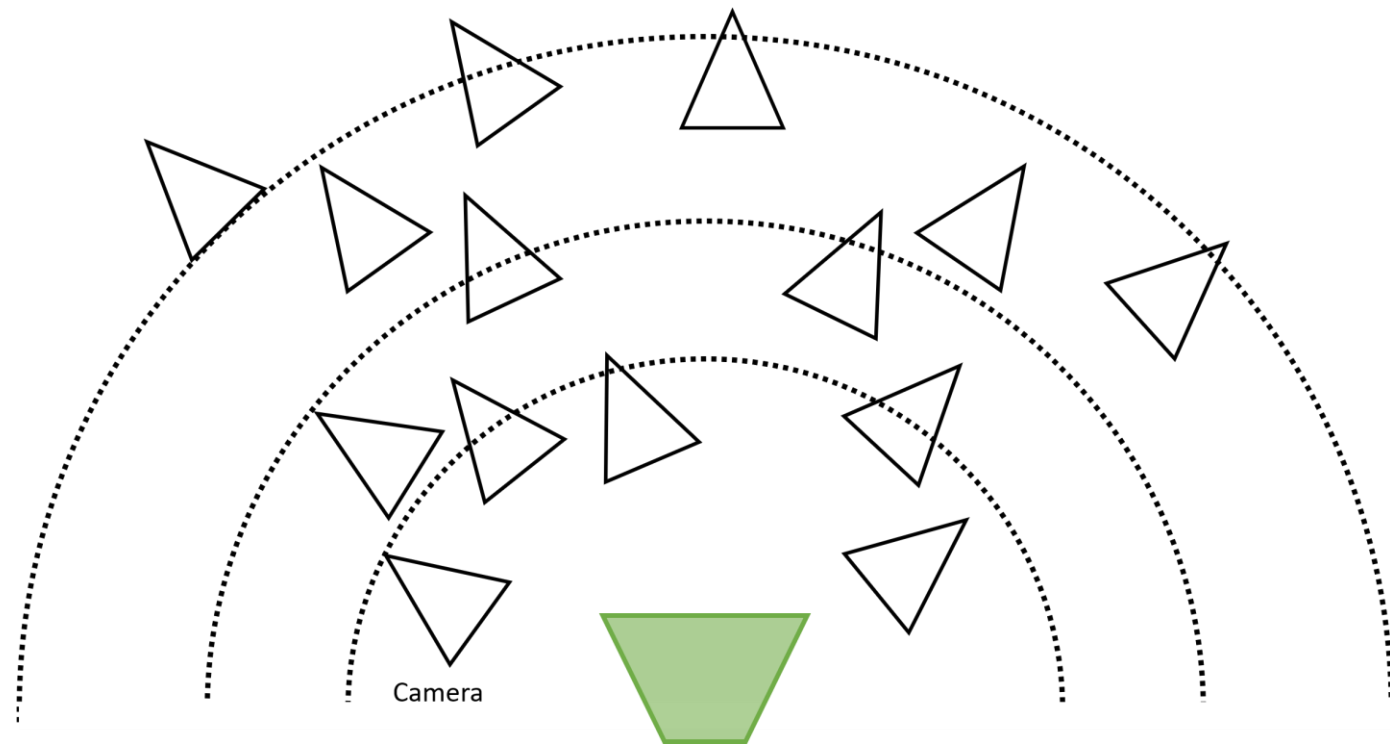


Image dataset B

RGB-D 画像のレンダリング

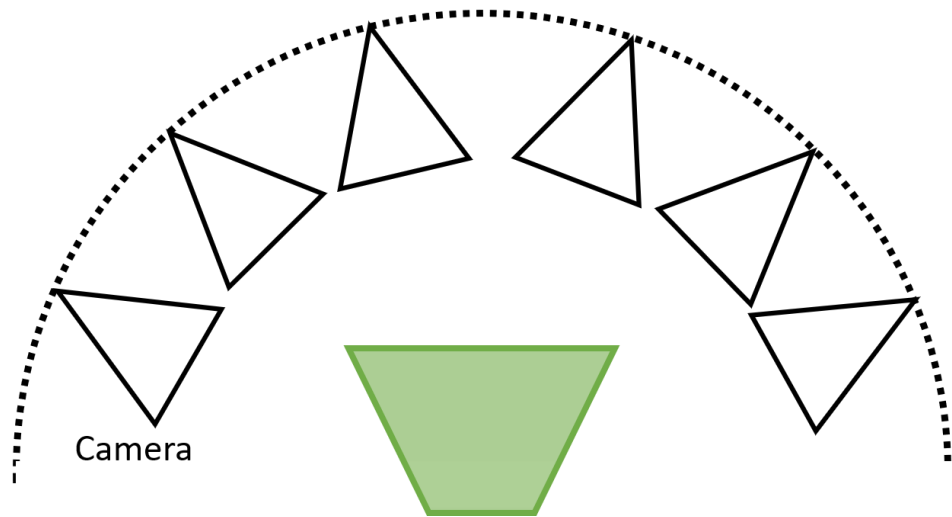


Image dataset A

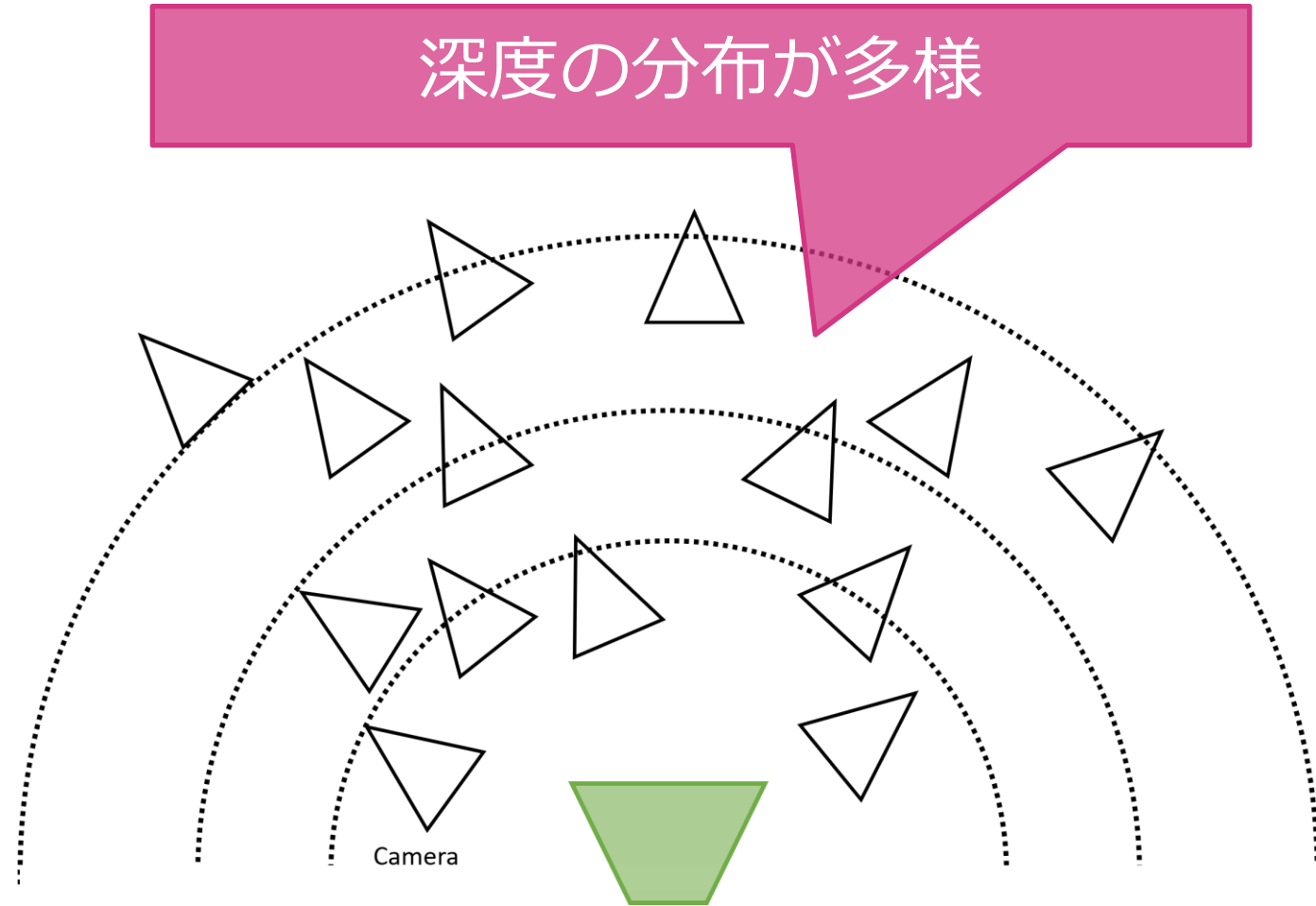


Image dataset B

- エンコーダは3種類。
 - Custom UNet : UNet likeな**自作ネットワーク**
 - ResNet50 (Layer 4) : ResNet50 の**最終レイヤー**の特徴量を用いる
 - ResNet50 (Layer 1-4) : ResNet50 の4つの**中間レイヤー**の特徴量を用いる

backbone	output
Custom UNet	$128 \times 112 \times 112$
ResNet50 (Layer 1)	$255 \times 56 \times 56$
ResNet50 (Layer 2)	$512 \times 28 \times 28$
ResNet50 (Layer 3)	$1024 \times 14 \times 14$
ResNet50 (Layer 4)	$2048 \times 7 \times 7$

入力は全て
 $3 \times 224 \times 224$

- エンコーダは3種類。
 - Custom UNet : UNet likeな**自作ネットワーク**
 - ResNet50 (Layer 4) : ResNet50 の**最終レイヤー**の特徴量を用いる
 - ResNet50 (Layer 1-4) : ResNet50 の4つの**中間レイヤー**の特徴量を用いる

backbone	output
Custom UNet	128 × <u>112 × 112</u>
ResNet50 (Layer 1)	255 × 56 × 56
ResNet50 (Layer 2)	512 × 28 × 28
ResNet50 (Layer 3)	1024 × 14 × 14
ResNet50 (Layer 4)	2048 × 7 × 7

入力は全て
3 × 224 × 224

- エンコーダは3種類。
 - Custom UNet : UNet likeな**自作ネットワーク**
 - ResNet50 (Layer 4) : ResNet50 の**最終レイヤー**の特徴量を用いる
 - ResNet50 (Layer 1-4) : ResNet50 の4つの**中間レイヤー**の特徴量を用いる

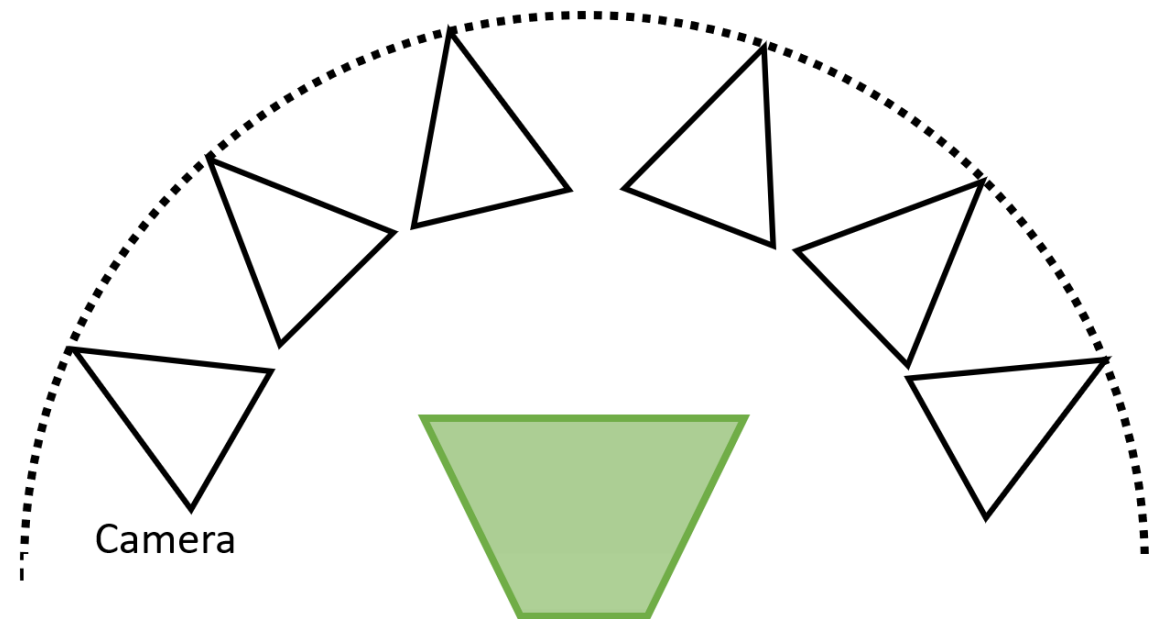
backbone	output
Custom UNet	128 × <u>112 × 112</u>
ResNet50 (Layer 1)	255 × 56 × 56
ResNet50 (Layer 2)	512 × 28 × 28
ResNet50 (Layer 3)	1024 × 14 × 14
ResNet50 (Layer 4)	<u>2048</u> × 7 × 7

入力は全て
3 × 224 × 224

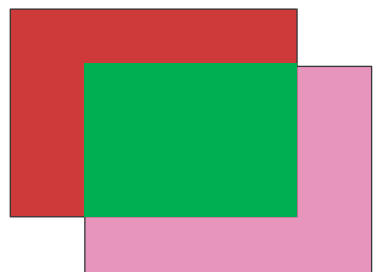
内容

① 食事と食器の三次元再構成

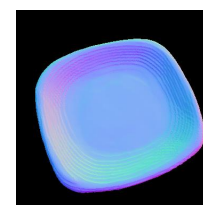
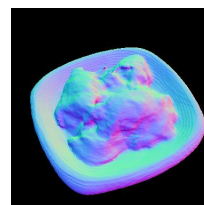
- **エンコーダ**の比較



食事と食器の三次元再構成：エンコーダの比較




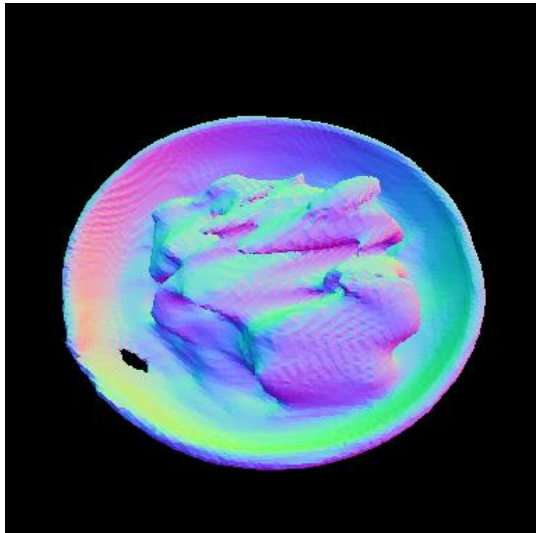
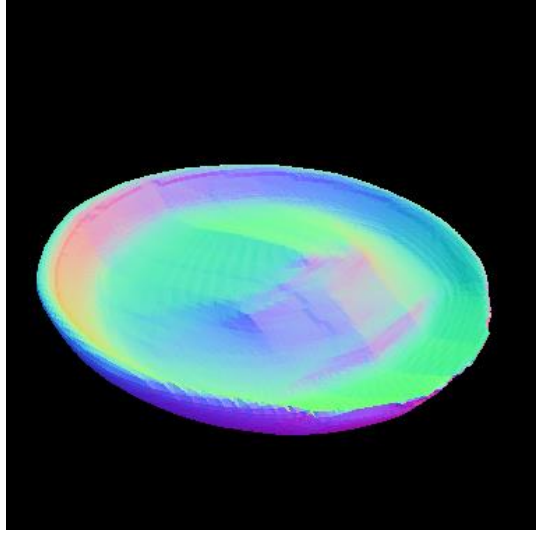
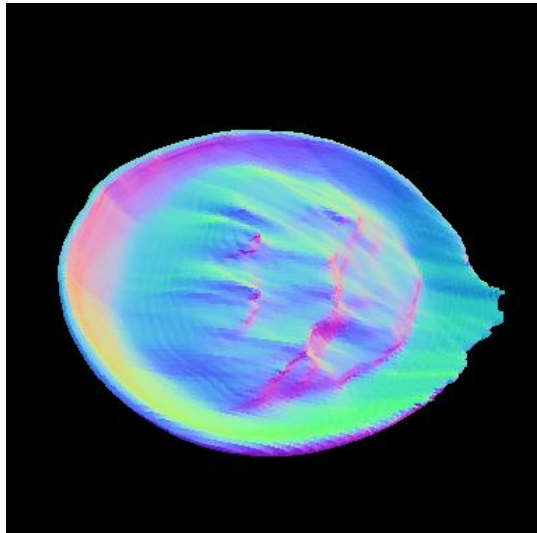

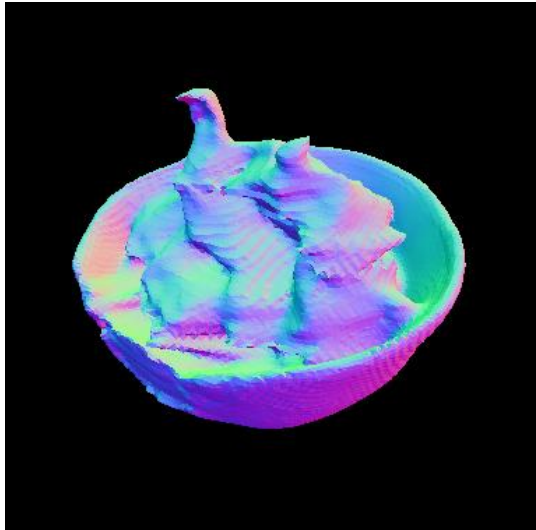
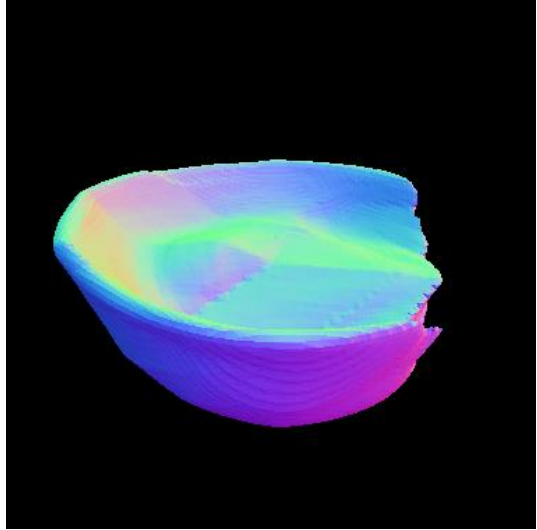
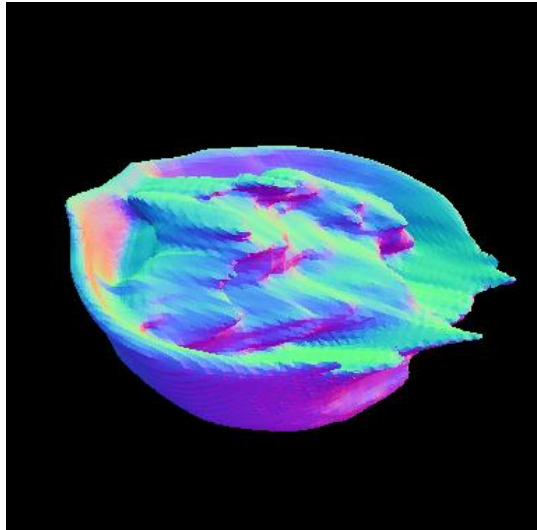
$$\text{IoU} = \frac{\text{重なり率}}{\text{重なり率}}$$




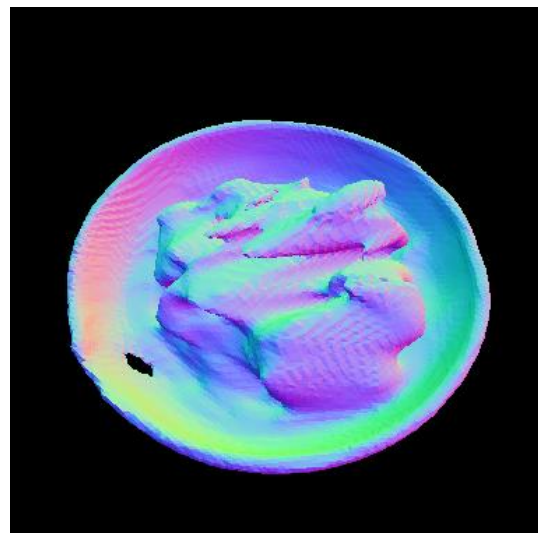
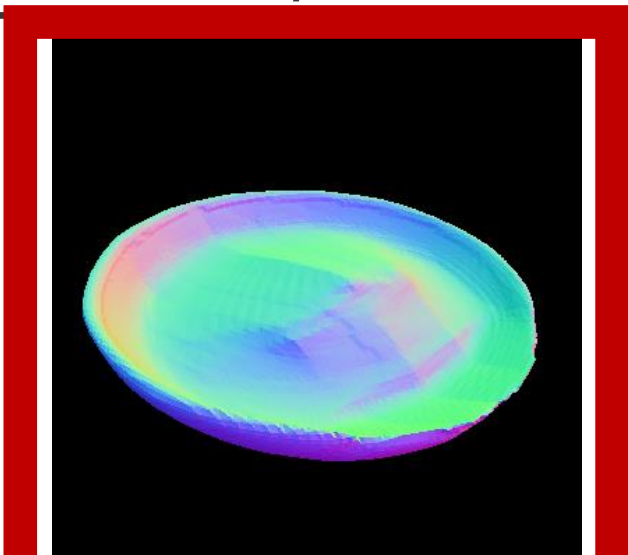
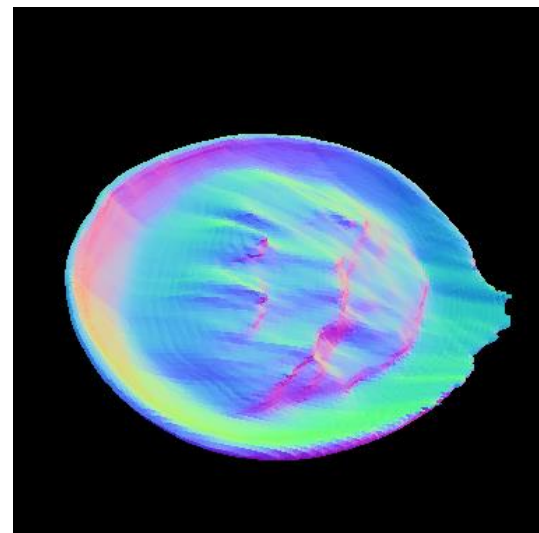

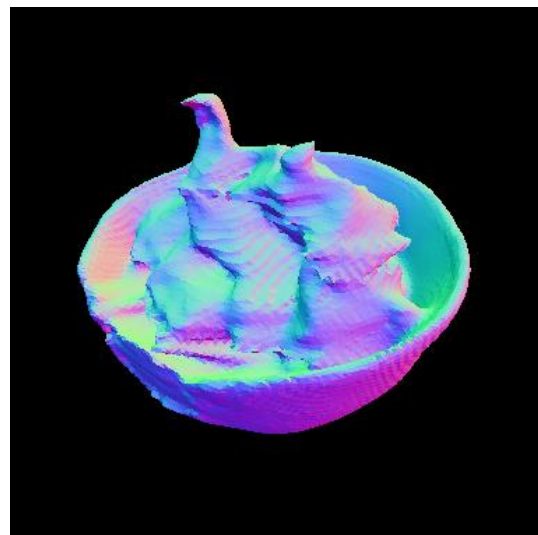
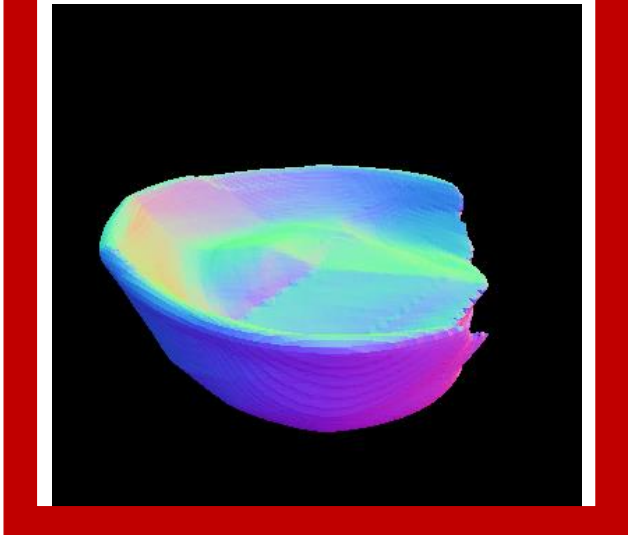
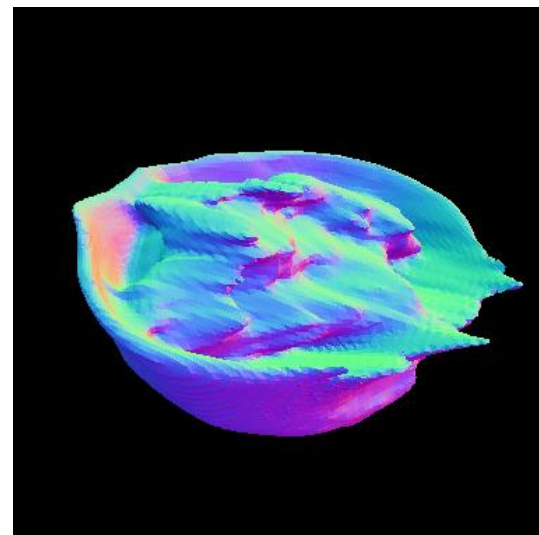
dish — plate = food (食品)

encoder	IoU (dish)↑	IoU (plate)↑	食品体積 誤差 (cm ³)↓	相対食品 体積誤差↓
Custom UNet	0.702	0.534	46.046	13.0 %
ResNet50 Layer 4	0.636	0.437	99.129	37.7 %
ResNet50 Layer 1-4	0.558	0.470	54.293	16.6 %


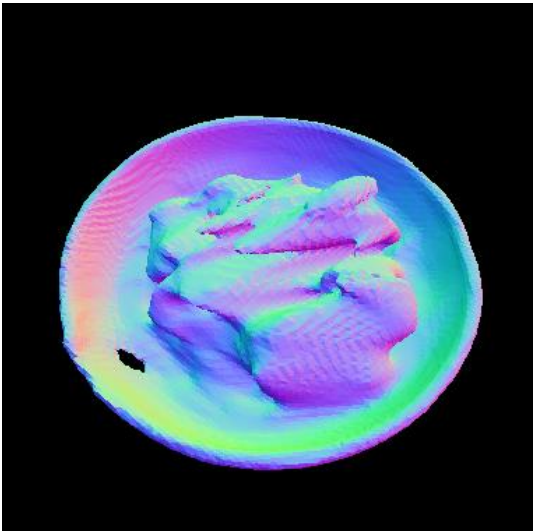
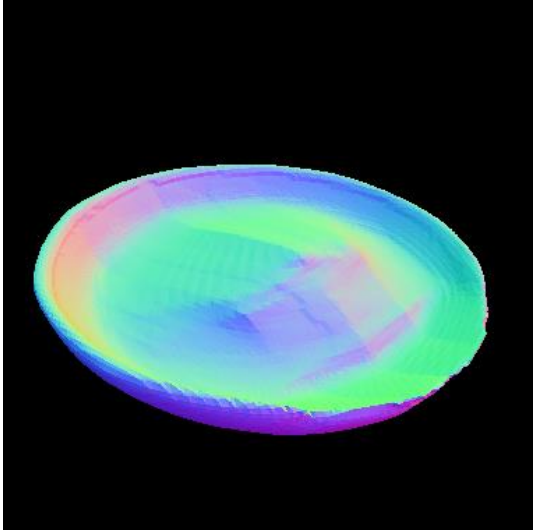
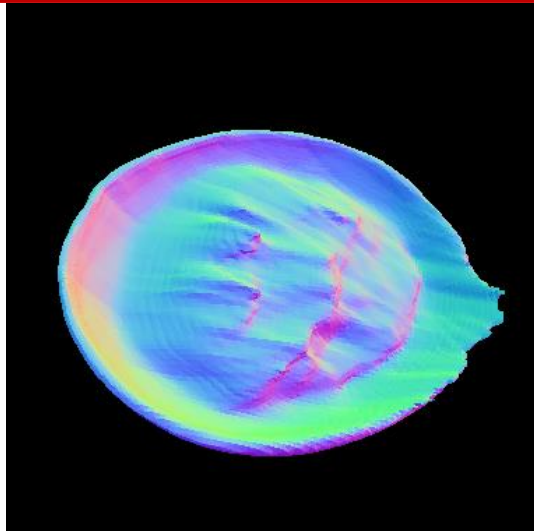

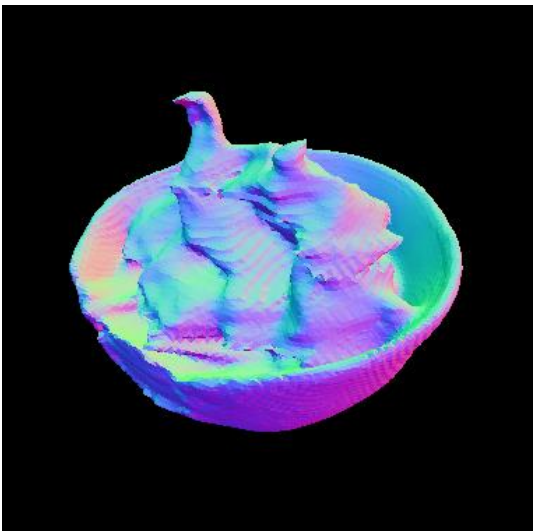
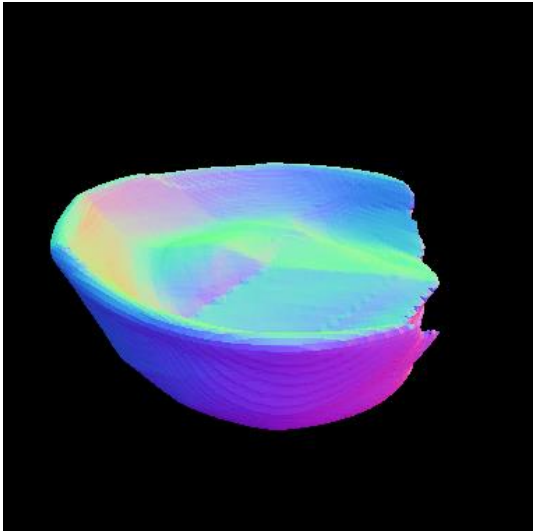
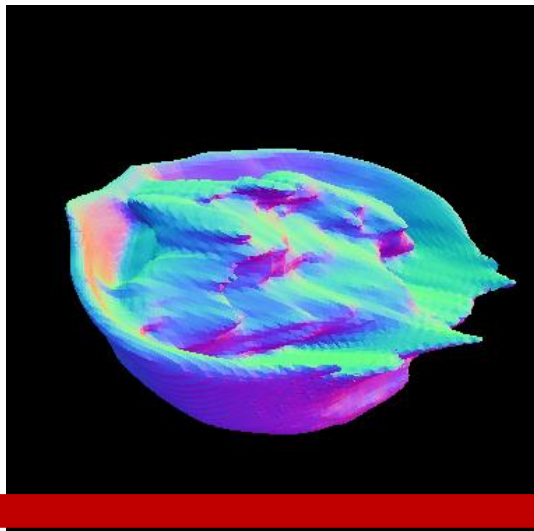
食事と食器の三次元再構成

Method Input	Custom UNet	ResNet50 Layer4	ResNet50 Layer1-4
			
			


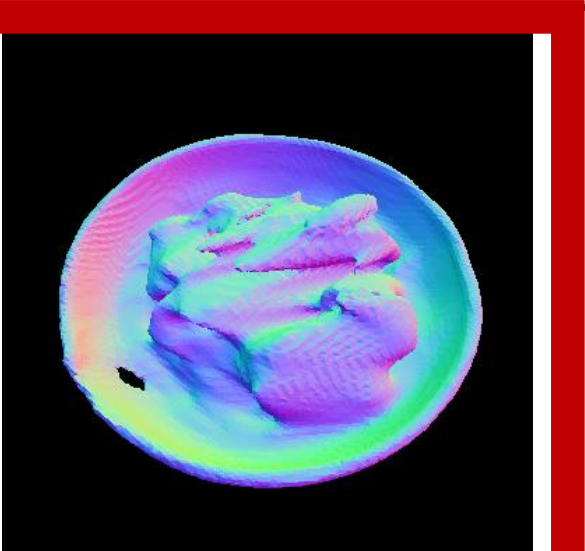
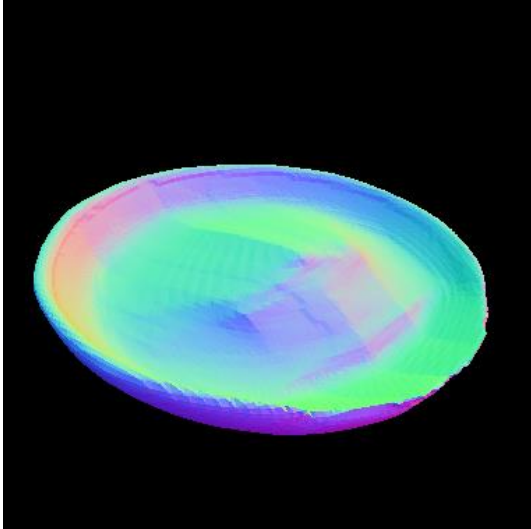
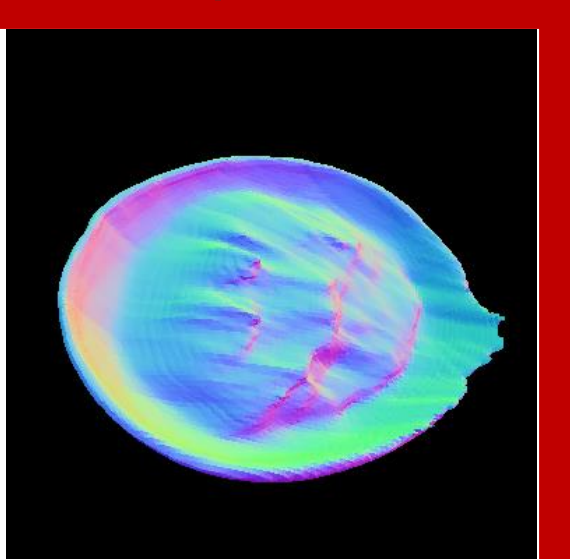

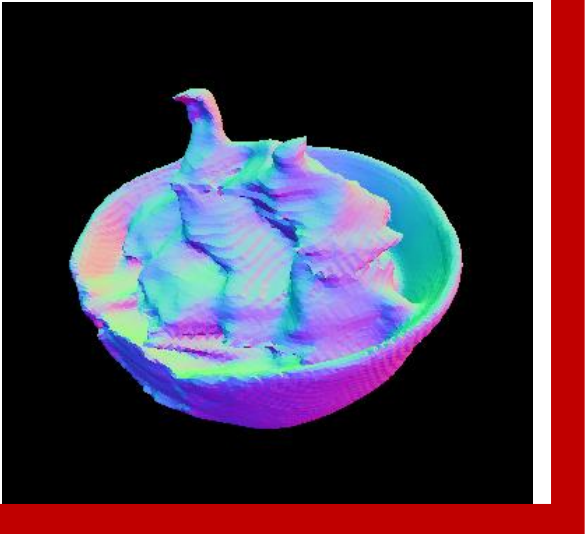
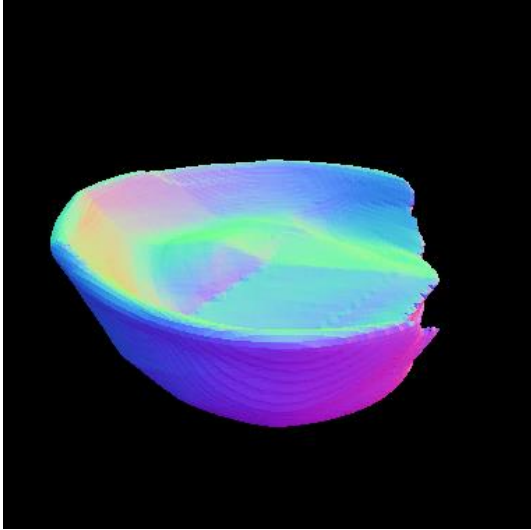
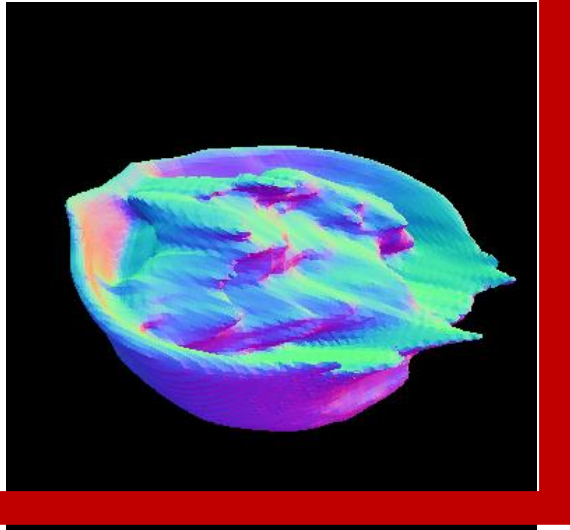
食事と食器の三次元再構成

Method Input	Custom UNet	ResNet50 Layer4	ResNet50 Layer1-4
			
			

食事と食器の三次元再構成

Method Input	Custom UNet	ResNet50 Layer4	ResNet50 Layer1-4
			
			

食事と食器の三次元再構成

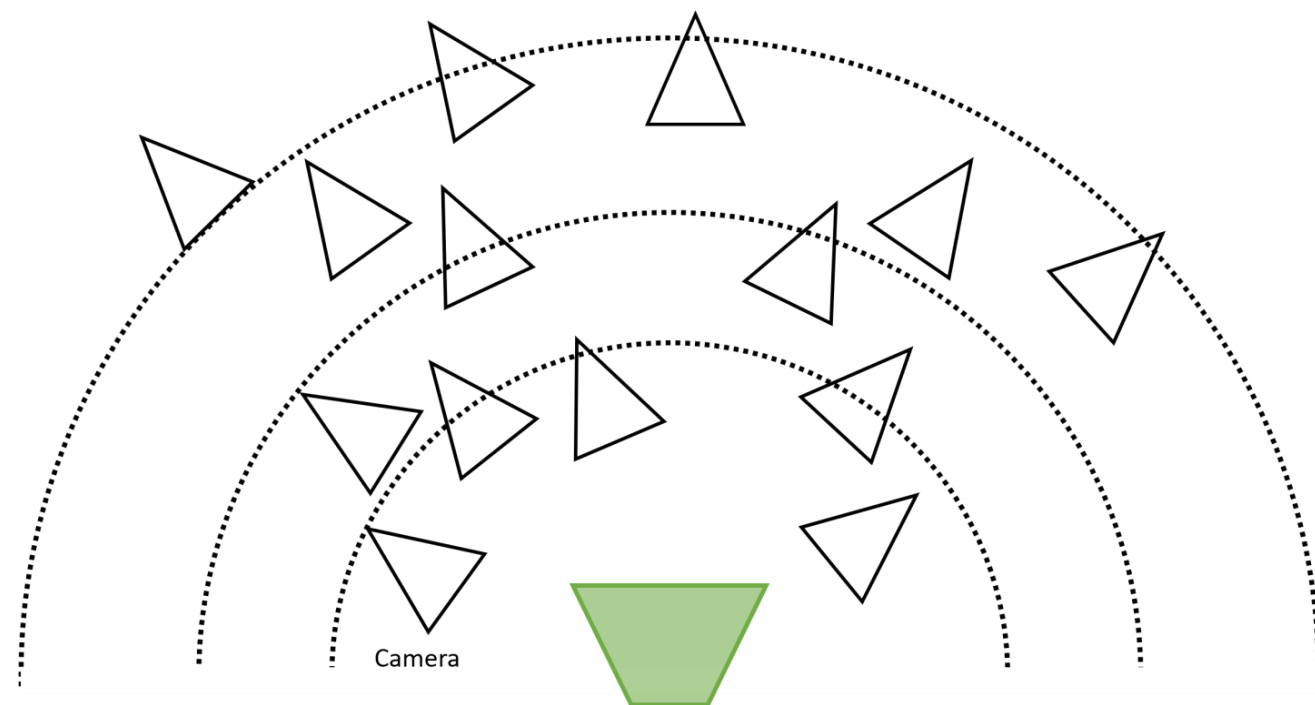
Method Input	Custom UNet	ResNet50 Layer4	ResNet50 Layer1-4
			
			

②4種類の深度画像の活用手法の比較

- 深度値サンプリング (S)
- CNNで特徴量抽出 (C)
- S + C
- 利用しない(None)

エンコーダには先ほどの実験で
もっと良い精度だった

Custom UNet


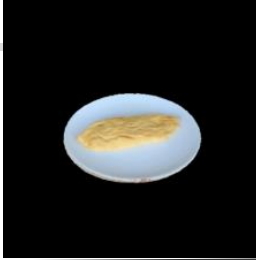

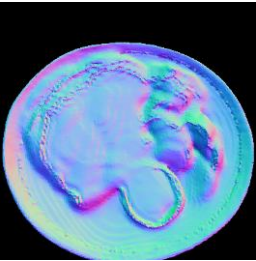
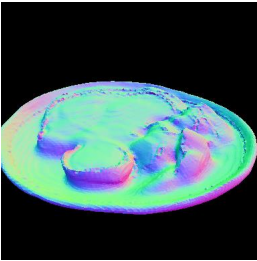
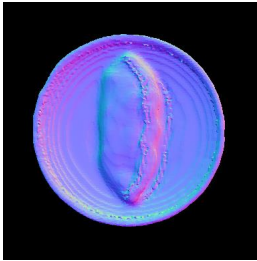
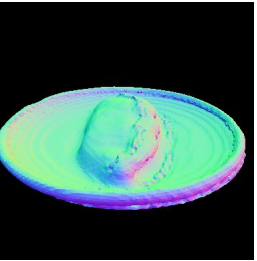
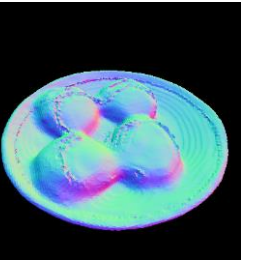
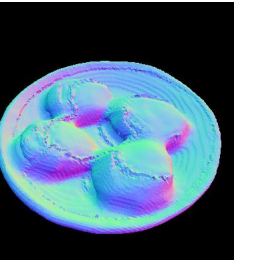
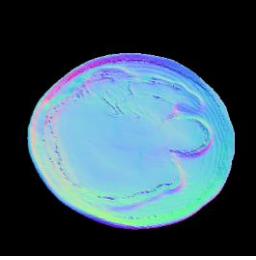
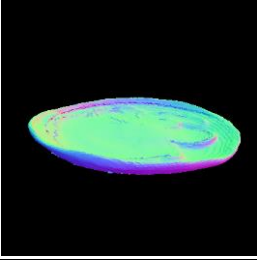
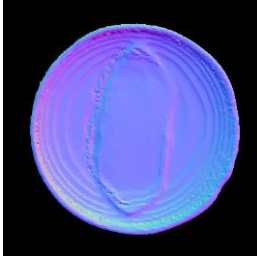
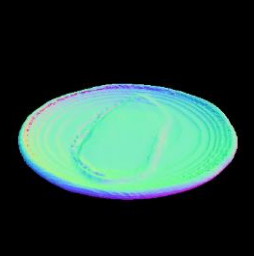
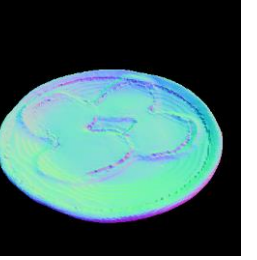
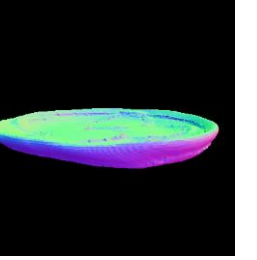
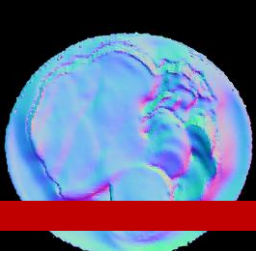
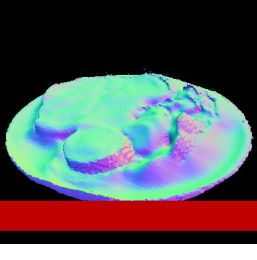
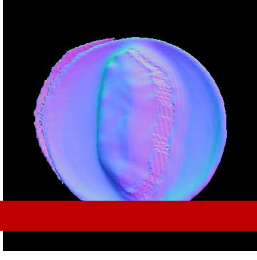
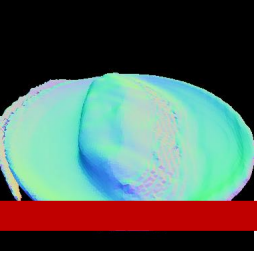
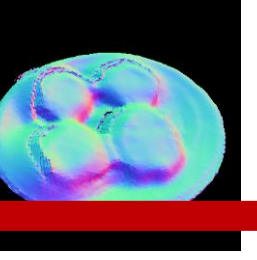
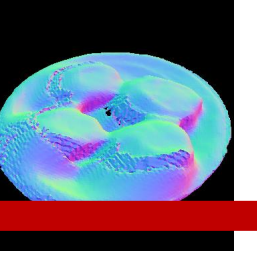
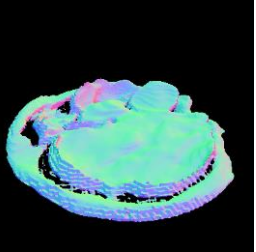
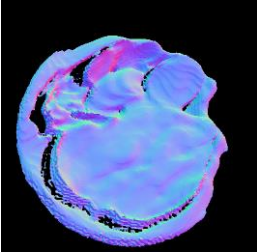
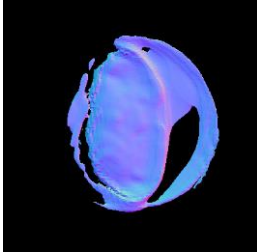
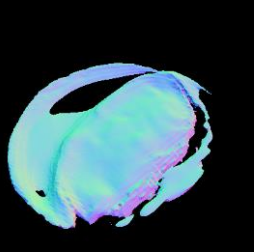
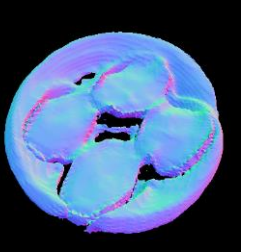
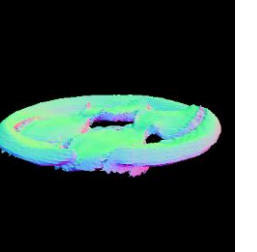


4種類の深度画像の活用手法の比較

- 定量的結果
 - S: 深度値サンプリング
 - C: 深度特徴量 (CNN)

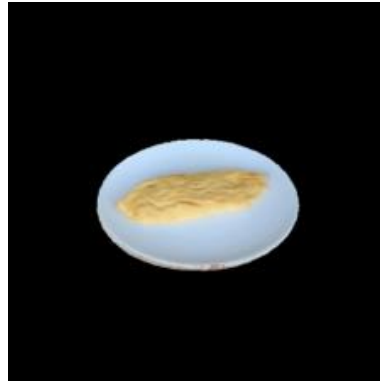
Depth	Valid	IoU (dish)↑	IoU (Plate)↑	食品体積 (cm ³)↓	相対食品 体積誤差↓
C + S	24/24	0.567	0.407	51.24	15.0 %
C	24/24	0.534	0.337	90.291	25.9 %
S	23/24	0.365	0.124	104.122	32.0 %
None	1/24	invalid	invalid	invalid	invalid

4種類の深度画像の活用手法の比較

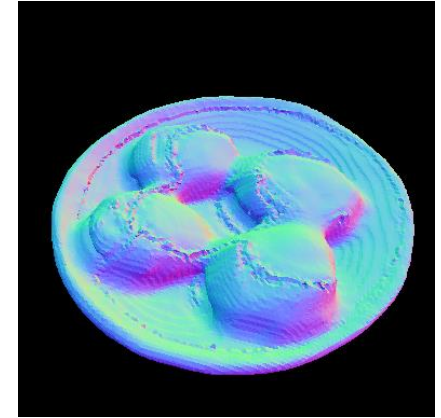
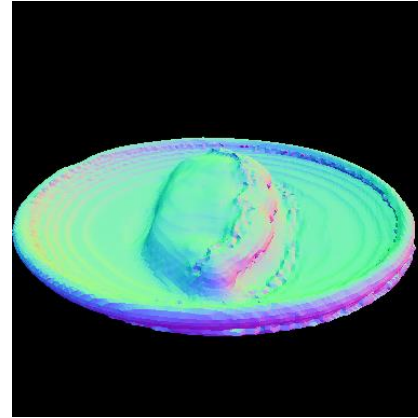
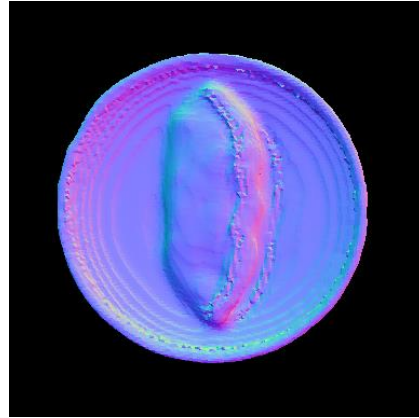
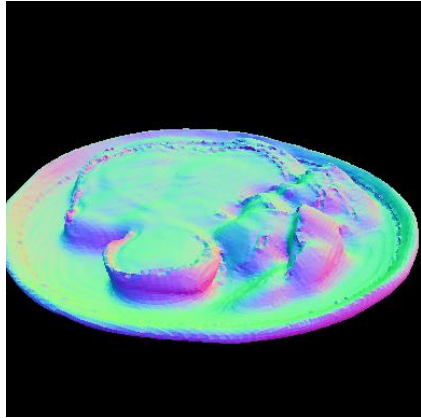
		Input						
Method								
Custom UNet (S + C)	Dish							
	Plate							
	Dish							
	Plate							
	Dish							
	Plate							

4種類の深度画像の活用手法の比較

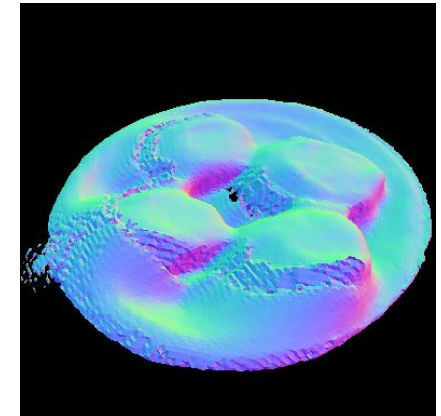
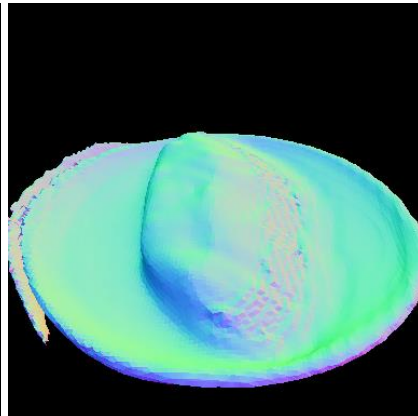
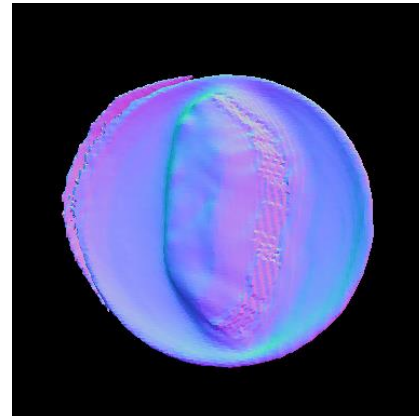
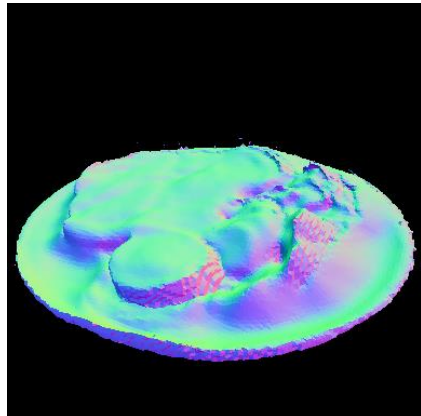
Input



Custom UNet (S + C)

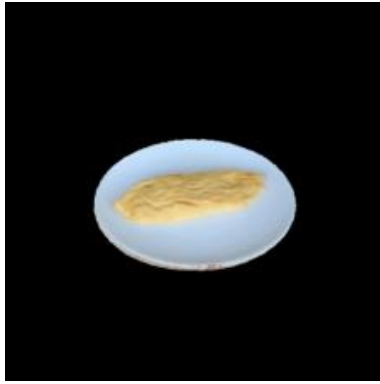


Custom UNet (C)

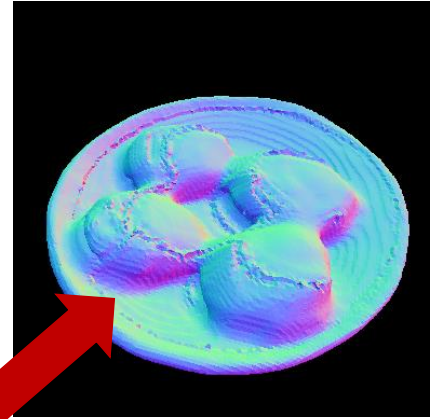
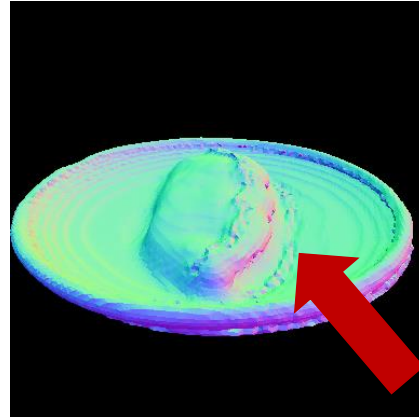
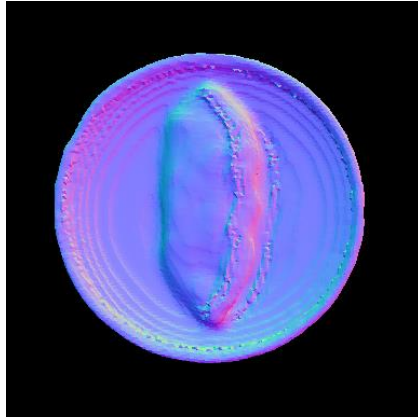
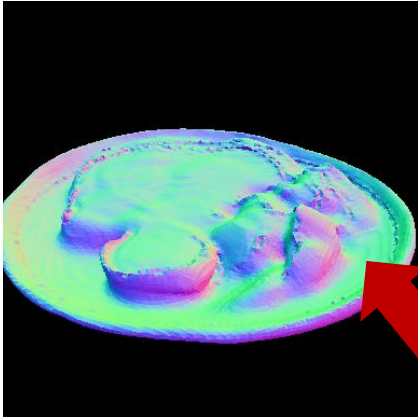


4種類の深度画像の活用手法の比較

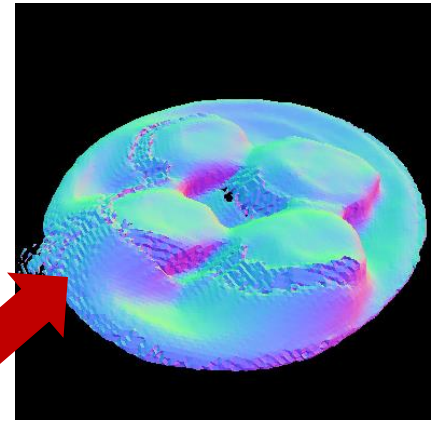
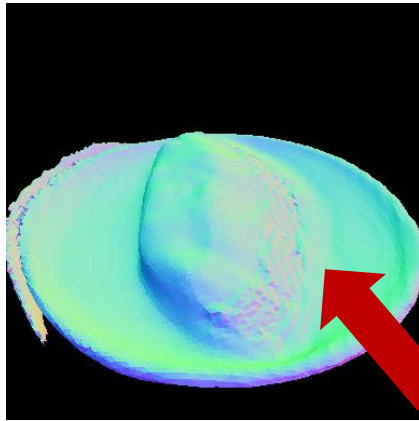
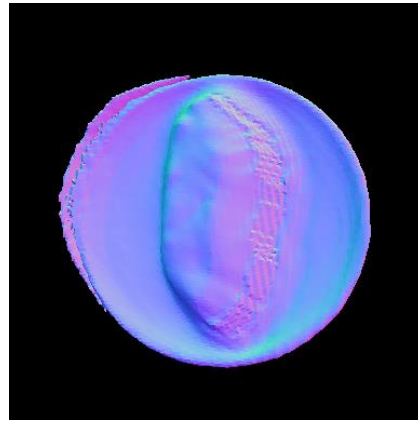
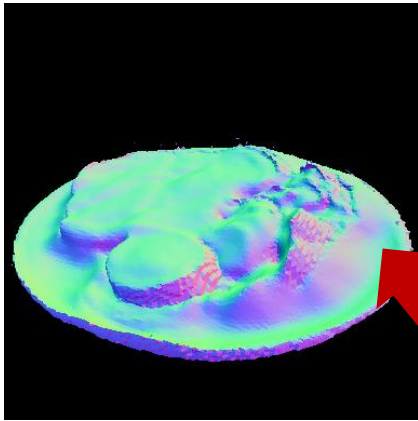
Input



Custom UNet (S + C)



Custom UNet (C)



- **単一 RGB-D 画像 からの食事と食器の実寸三次元再構成と体積推定**
 - **正規化を行わず**に、**実寸**通りの三次元形状を再構成
 - 新しい深度画像の活用手法
- 高精度な再構成 + **実体積**推定を実現