Real Scale Hungry Networks: Real Scale 3D Reconstruction of a Dish and a Plate using Implicit Function and a Single RGB-D Image

Shu Naritomi*
naritomi-s@mm.inf.uec.ac.jp
NTT Communications Corporation, Japan

Keiji Yanai
yanai@cs.uec.ac.jp
The University of Electro-Communications, Tokyo, Japan

ABSTRACT

The management of dietary calorie content using information technology has become an essential topic in the multimedia field of research in recent years. Therefore, many researchers and companies are conducting research and developing applications. Many methods for estimating the calorie content of a food use image recognition. However, these methods have a problem. They cannot consider the 3D heights and depths of the food because they only consider the food as a 2D object, even though the actual meal is 3D. To solve this problem, we would like to utilize 3D reconstruction techniques based on deep learning, developed in recent years, but most of these methods reconstruct the normalized objects. Being normalized means that the actual size is unknown, making it difficult to use them for estimating calories and nutritional value. In this paper, we propose a method using an implicit function representation that reconstructs the 3D shapes of a dish and plate as they are in real scale, using an RGB-D image and camera parameters.

CCS CONCEPTS

• **Computing methodologies → Scene understanding**.

KEYWORDS

food volume estimation, 3D reconstruction from a single image, food image recognition

ACM Reference Format:
Shu Naritomi and Keiji Yanai. 2022. Real Scale Hungry Networks: Real Scale 3D Reconstruction of a Dish and a Plate using Implicit Function and a Single RGB-D Image. In *Proceedings of the 7th International Workshop on Multimedia Assisted Dietary Management (MADiMa '22), October 10, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3552484.3555750

*This work was done in his former affiliation, The University of Electro-Communications, Tokyo.

MADiMa '22, October 10, 2022, Lisboa, Portugal
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9502-1/22/10…$15.00
https://doi.org/10.1145/3552484.3555750

1 INTRODUCTION

In recent years, the management of dietary calorie content using information technology has been an essential topic. As a result, various methods and applications for estimating calorie content have been researched and developed. Most of the existing methods for estimating the calorie content of foods use image recognition [4, 5]. Therefore, many methods only recognize food in 2D, even though actual food is 3D objects. Although there are methods that recognize food in three dimensions [1, 11], they cannot recognize a bowl of rice because of the restriction that the food must be on a flat plate. To solve this problem, previously, we proposed a method called "Hungry Networks" [12] which reconstructs a dish (food + plate) and plate in 3D from a single RGB image using an implicit function representation and achieves highly accurate reconstruction and volume estimation.

However, Hungry Networks has one problem. That is, the reconstructed 3D shapes are normalized. Since the actual volume cannot be determined from the normalized 3D shape alone, measuring the actual size separately from the reconstruction was necessary. To solve this problem, we propose a method using an implicit function representation that reconstructs the actual 3D shape without normalization by utilizing an RGB-D image and a perspective projection model, which is a classical camera model. The proposed method is called as "Real Scale Hungry Networks." The experimental results show that the proposed method can accurately reconstruct the 3D shape in real scale and can be used for actual volume estimation.

2 RELATED WORK

2.1 3D shape reconstruction from a single image

When performing 3D reconstruction using deep learning, it is important to decide what kind of representation is used for reconstruction. The representation can be mainly classified into voxel, point cloud, mesh, and volume. The voxel and point cloud representations methods [3, 6, 18] have problems such as high computational cost, inability to achieve high resolution, and complex post-processing. The mesh representations methods [13, 19] use a mesh template or dynamically generate a mesh template and optimize the mesh shape. Compared to voxel-based methods, mesh representation has many advantages, such as high resolution and memory efficiency. Also, unlike point clouds, they can represent shape because it has connection information between points. However, mesh template-based methods have not achieved good results because they deform the template mesh to obtain the target 3D shape and thus have problems such as low expressive power and self-intersections. In recent years, volume representation using implicit functions has been attracting attention. The volume representation method with implicit functions [10, 12, 15] learns a function representing a 3D shape as a scalar field. Finally, the 3D shape is extracted as a mesh by applying a Marching Cube to the inferred scalar field. This implicit function representation is revolutionary because it is far more expressive than previous methods, achieves highly accurate reconstruction, and is superior in memory efficiency and network size.

2.2 3D reconstruction using depth images

There are two types of 3D reconstruction methods using depth images: those that use only depth images [16] and those that use both RGB and depth images [8]. In these methods using depth images, the depth images are converted to a volumetric grid representation with or without RGB images and are used as inputs to the network. The conversion to a volumetric grid representation is compatible with 3D Convolution and voxel representation, but 3D Convolution is computationally expensive. Moreover, voxel representation has a problem of low expressive power and resolution compared to implicit function representation. Therefore, in this paper, we propose a reconstruction method using implicit function representation that integrates feature tensors of depth images and RGB images without converting them to volumetric grid representation.

2.3 Food recognition considering 3D shape

Chen et al. [2] use depth sensors to capture depth images and estimate the amount of calories in a meal. There are also methods to recover 3D shapes by estimating camera matrices from multiple viewpoints, such as the method by Puri et al. [14] and DietCam [7]. In recent years, research using CNNs has been developed. Lu et al. [9] generated depth images using deep learning and attempted to infer the food volume from the generated depth images. Im2calories [11] estimates 3D shapes from RGB images in the form of voxel representation and utilizes them for calorie estimation. Recently, a dataset of 5000 RGB-D images annotated with the nutritional value, named Nutrition5k [17] has been published.

3 METHOD

The proposed method named "Real Scale Hungry Networks" reconstructs two non-normalized, real-scale 3D shapes of a dish and a plate using an RGB-D image and camera parameters as input. The unique feature of this method is that the occupancy field is inferred not in the normalized space but the space corresponding to the real scale. This feature solves the problem of our previous work, "Hungry Networks" [12]; that is, the reconstructed 3D shape is normalized so that the real scale must be calculated by a different method from the reconstruction. In order to achieve this, we propose a method that abandons the setting of 3D reconstruction in a normalized space, which is easy to learn for deep learning and utilizes a perspective projection model, which is one of the classical camera models, and depth images.

3.1 Camera Models and Deep Learning

In general, using a perspective projection model and a depth image, it is possible to calculate the real scale of objects in an image. However, only the depth image cannot calculate the volume of an object because it does not provide the shape of backside. Therefore, it is necessary to perform 3D reconstruction by deep learning to

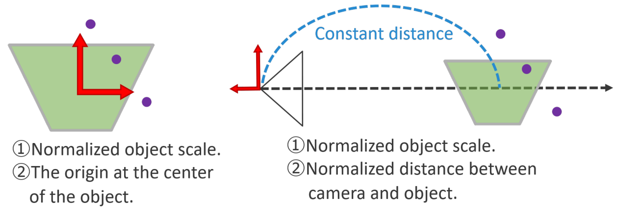
①Normalized object scale.
②The origin at the center of the object.
Constant distance
①Normalized object scale.
②Normalized distance between camera and object.

Figure 1: left: Normalization performed by most implicit function representation-based methods, right: Normalization performed by PIFu [15]

obtain the complete 3D shape. Also, we would like to reconstruct the real scale 3D shape without normalization by utilizing the perspective projection model and depth images. However, as shown in the left panel of Figure 1, the 3D reconstruction method using implicit function representation sets the origin at the center of the object and performs normalization to keep the object's size constant. Therefore, it is difficult to integrate it with the camera model. Hence, we focus on a method named PIFu [15], which performs 3D reconstruction from a single RGB image using an implicit function representation utilizing the camera model. This method uses the weak perspective projection method for the camera model to normalize the object size and the distance from the camera to the object to be reconstructed, as shown in the right panel of Figure1. While the normalization by using the weak perspective projection method made it possible to learn, the essential information, such as the actual depth and the actual scale of the object, is destroyed. Thus the reconstruction cannot be performed as it is in actual size. There is also the problem that only certain viewpoints can be reconstructed correctly in PIFu.

The problem to be solved in this research is not to reconstruct normalized 3D shapes but to reconstruct 3D shapes in real scale with high accuracy. At the same time, the system must be able to handle images from arbitrary viewpoints as input. The application will not be helpful if it can only handle input from specific viewpoints. Therefore, the method proposed in this paper reconstructs a real size 3D shape using a perspective projection rather than a weak perspective projection. However, when using the perspective projection, the depth of the space to be reconstructed cannot be normalized as in the weak perspective projection, making learning difficult. Hence, depth images are also used as input in the proposed method in addition to RGB images. In this method, the depth image is used as follows. (1) Setting the space for which the occupancy field should be inferred, (2) Sampling the distance to the visible surface from the camera, (3) Extraction of features related to global shape. These methods make it possible to train the network that realizes 3D reconstruction according to the actual scale, and the reconstruction accuracy is also improved.

3.2 Utilization of depth images

In this section, we propose methods for highly accurate 3D reconstruction corresponding to actual scale using depth images.

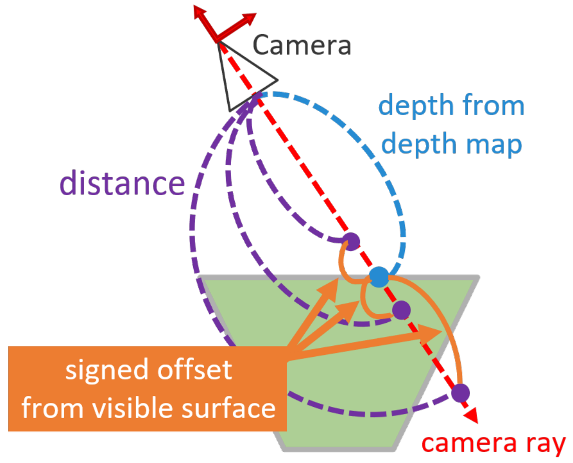

Figure 2: A signed offset from the surface of the object can be calculated from the difference between the depth value obtained from the pixel values in the depth image and the distance to the point for which the occupancy is desired. This offset falls within a specific range with respect to the object's surface. Therefore, the object shape can be learned from the offset without normalizing the depth value.

3.2.1 Decision on the space for inferring the occupancy field. In order to obtain the actual 3D shape, the space for finding the occupancy field must first be set correctly. When reconstructing a normalized 3D shape as before, it is sufficient to sample the coordinates from the normalized space. For example, we can restrict the space between the x, y, z axes to [-0.5, 0.5]. However, we do not know the space to be sampled when using perspective projection. This is because the viewing platform extending from the camera has no restrictions on the depth direction, and it is impossible to know where the object is. This problem can be solved by using a depth image. Because the pixel value of the depth image indicates how in-depth the meal is, and based on that, the space for which the occupancy field should be obtained can be set.

3.2.2 Depth value sampling from depth image. Learning was relatively easy in reconstructing normalized 3D shapes because it only had to infer the occupancy field in constant space. Also, PIFu learned well by using weak perspective projection, which normalizes the depth at which objects exist within a specific range. However, when 3D shapes are handled as an actual scale without normalization, as in this research, it is difficult to train the network because the distance between the camera and the object is quite different. So we utilize the depth value obtained from the depth image. We focused on the difference between the depth value taken from the depth image pixel and the distance from the camera to the point p for which we wanted to infer the occupancy. This difference is a signed offset, which indicates how far the point $p \in \mathbb{R}^3$ is from the object's surface visible to the camera. Since this offset is within a specific range for the object's surface, we thought that the network might be able to learn the 3D reconstruction correctly based on the offset without normalizing the depth values. This is shown in Figure 2. Therefore, we decided to sample depth values from the depth image and use them as input to the decoder. We call this depth value sampling. We will show in later experiments whether this idea contributes to the accuracy.

3.2.3 Depth features for overall shape. In depth value sampling, information was obtained at the pixel level from the depth image. However, this alone misses beneficial information on the surface shape around the pixel and the entire 3D object's shape. Therefore, we decided to apply CNN to depth images as well as RGB images to extract features and utilize them. We call them depth features.

3.2.4 Network. The network overview of the proposed method is shown in Figure 3. The network consists of two encoders and two decoders. Two encoders exist to obtain features for each RGB image and depth image. There are also two decoders to infer the occupancy of each dish and plate.

3.2.5 Inference. This section describes the behavior during inference. First, the point $p \in \mathbb{R}^3$ is sampled, and the coordinates $(u, v) \in \mathbb{R}^2$ of the point p projected on the image are calculated using the perspective projection model. Next, the encoders are used to extract the features of the RGB image and the depth image, respectively. Here, the shapes of both features are $C_1 \times H \times W$, and $C_2 \times H \times W$, respectively. These features need to be computed only once at the beginning of inference. Next, we infer the occupancy rate. For example, consider inferring the occupancy of the 3D coordinate $p \in \mathbb{R}^3$ indicated by the purple dots in Figure 2. For inference, we use the features s, t, d, which are bilinear interpolation sampled from the RGB features/depth features/depth images using (u, v), and z, which is the distance from the camera to the point p. The shapes of the features, s, t, d, z, are $C_1 \times 1 \times 1$, $C_2 \times 1 \times 1$, $1 \times 1 \times 1$, and $1 \times 1 \times 1$, respectively. These four features are combined in a concatenate layer to create the features which shape is $(C_1 + C_2 + 2) \times 1 \times 1$. The created features are used as inputs to two decoders, each of which performs 1D convolution to infer occupancy.

3.2.6 Train. The mini-batch loss for training the network is defined as follows:

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & | & \mathbf{T} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

$$\begin{aligned}
\boldsymbol{e}_i &= \text{encoder}_{rgb}(I_i) && (2)\\
\boldsymbol{f}_i &= \text{encoder}_{depth}(D_i) && (3)\\
(u,v)_{i,j} &= \text{projection}(p_{i,j}, K_i, R_i, T_i) && (4)\\
\boldsymbol{s}_{i,j} &= \text{sample}(\boldsymbol{e}_i, (u,v)_{i,j}) && (5)\\
\boldsymbol{t}_{i,j} &= \text{sample}(\boldsymbol{f}_i, (u,v)_{i,j}) && (6)\\
\boldsymbol{d}_{i,j} &= \text{sample}(D_i, (u,v)_{i,j}) && (7)\\
\boldsymbol{z}_{i,j} &= \text{distance}(p_{i,j}, K_i, R_i, T_i) && (8)\\
\boldsymbol{c}_{i,j} &= \text{concatenate}(\boldsymbol{s}_{i,j}, \boldsymbol{t}_{i,j}, \boldsymbol{d}_{i,j}, \boldsymbol{z}_{i,j}) && (9)\\
y1_{i,j} &= \text{decoder}_{dish}(\boldsymbol{c}_{i,j}) && (10)\\
y2_{i,j} &= \text{decoder}_{plate}(\boldsymbol{c}_{i,j}) && (11)
\end{aligned}$$

$$\mathcal{L}_O(\hat{o}, o) = \mathcal{L}_{bce}(\hat{o}, o) \tag{12}$$

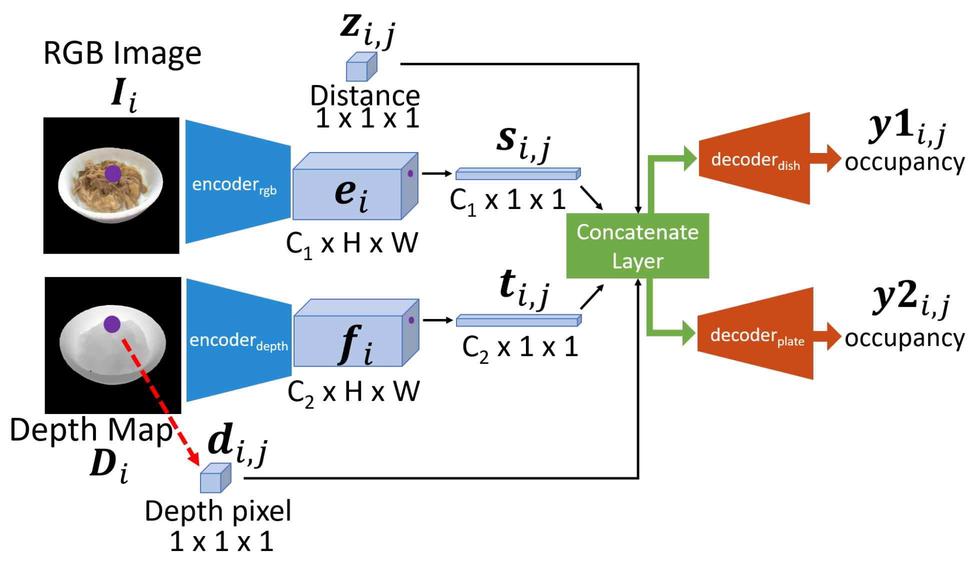

Figure 3: The network consists of two encoders and two decoders. The encoders extract features from RGB and depth images, respectively. The decoders infer the occupancy of a dish and a plate, respectively.

$$\mathcal{L}_C(o1, o2) = \max(o2 - o1, 0) \tag{13}$$

$$\mathcal{L}_\mathcal{B} = \frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\sum_{j=1}^{K}\Big(\lambda_1\mathcal{L}_O(y1_{i,j}, o1_i(p_{i,j})) + \lambda_2\mathcal{L}_O(y2_{i,j}, o2_i(p_{i,j})) + \lambda_3\mathcal{L}_C(y1_{i,j}, y2_{i,j})\Big) \quad , \tag{14}$$

where **K** in Eq.1 is the intrinsic camera parameter matrix of the 3×3, and **R**, **T** are the extrinsic camera parameter matrices representing the 3×3 and 3×1 rotations and translations, respectively. The u, v on the left side are the coordinates of the point $p = (x, y, z)$ projected onto the image. In Eq.2,3, I_i and D_i are the ith RGB/depth images of the mini-batch, encoder$_{rgb}$, encoder$_{depth}$ are the encoders that extract features for each RGB and Depth, and $\boldsymbol{e}_i$, $\boldsymbol{f}_i$ are the extracted features. The projection in Eq.4 is a function to get u, v on the left side calculated using the right side of Eq. 1, and sample in Eq.5,6,7 is a function to perform bilinear sampling using given features $\boldsymbol{e}_i$, $\boldsymbol{f}_i$ and $(u, v)_{i,j}$ from the depth image D_i, and $\boldsymbol{s}_{i,j}, \boldsymbol{t}_{i,j}, \boldsymbol{d}_{i,j}$ are the features extracted by bilinear sampling. Note that j means the j-th point out of K sampled points from the neighborhood of the i-th 3D shape of batch i. The distance in Eq.8 is the absolute value of the z-axis of the point $p_{i,j}$ in the camera coordinate system. Note that $\boldsymbol{z}_{i,j}$ is not the distance from the origin of the camera coordinate system to the point $p_{i,j}$, but the distance from the plane parallel to the projection plane passing through the origin of the camera. And decoder$_{dish}$ and decoder$_{plate}$ in Eq.10,11 are decoders for inferring dish and plate occupancy, respectively. Let $y1_{i,j}, y2_{i,j} \in \mathbb{R}$ be the inferred occupancy. The $\mathcal{L}_O$ in Eq.12 is the binary cross entropy loss for training occupancy, and the $\mathcal{L}_C$ in Eq.14 is the plate consistency loss [12] for maintaining consistency of the plates. The final $y1_{i,j}, y2_{i,j}$ and Eq. 14 are used for training. Note that $\boldsymbol{e}_i, \boldsymbol{f}_i, \boldsymbol{s}_{i,j}, \boldsymbol{t}_{i,j}, \boldsymbol{d}_{i,j}, \boldsymbol{z}_{i,j}, \boldsymbol{y1}_{i,j}, \boldsymbol{y2}_{i,j}$ in Eq.2~11 correspond to Figure 3.

3.3 Dataset

The dataset used for training/evaluation of the proposed method was generated using the watertight Mesh dataset used in Hungry networks. This dataset contains RGB images but not RGB-D images. Therefore, to train the proposed method, we rendered RGB-D images from the same mesh dataset as Hungry networks. Note that the watertight mesh data used for training Hungry networks is normalized, so we used it after restoring it to its actual size in the present method.

3.3.1 RGB-D image rendering. In order to train the proposed method, we created two datasets by rendering the dish mesh in two different ways. The differences between the two ways are shown in Figure 4. One is an RGB-D image rendered by sampling about 30 points from a hemispherical shape with a radius of 20 cm centered on the dish and pointing the camera at the meal from there. This way is close to the condition of the image rendered by Hungry Networks. The other is an RGB-D image rendered with the meal in the field of view from 25 sampled points each from a hemisphere of radius 20, 30, and 40 cm centered on the meal, to which noise sampled from a normal distribution with mean 0 and variance 2.5 cm was added. Compared to the first RGB-D image dataset, the second RGB-D image dataset has a different size of food in the image. Also, as Figure 4 shows, since the images are taken from various distances, the diversity of the depth images is higher than that of the first one, making learning more difficult. We refer to these two image

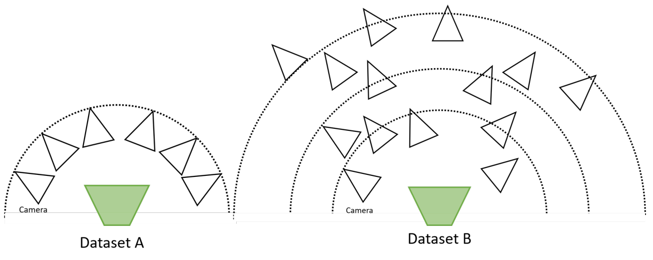

Figure 4: Image dataset A and B have different camera settings when rendering images. Image dataset B is more difficult to learn than Image dataset A because of the more diverse depth distribution.

datasets as Image dataset A and Image dataset B, respectively. Since the intrinsic/extrinsic parameters of the camera are essential in this method, the RGB-D images are captured, and the camera parameters are saved simultaneously. For implementation, we used the Open3D and OSMesa library for rendering.

4 EXPERIMENTS

The proposed method takes RGB-D images and camera parameters as input and performs a real-scale 3D reconstruction of a dish and plate. Since this method performs bilinear sampling from the feature tensors, we considered that the size of the feature tensors output by the encoder is important. Therefore, we first trained the method using various encoders and evaluated it quantitatively and qualitatively. For this experiment, we used Image dataset A, which is relatively easy to train.

Next, Image Dataset B was used to test how best to handle depth images to obtain the best accuracy. In other words, the experiment on the effects of depth value sampling and depth features. In this experiment, we used the encoder that had good accuracy in the first experiment.

4.1 Encoder

In this section, we experiment to see how different encoders affect the results. Three types of encoders were used for feature extraction: Custom UNet, ResNet50 Layer4, and ResNet50 Layer1-4. Custom UNet is a network of our own implementation with a Unet-like architecture. ResNet50 Layer4 uses the output of the final layer of ResNet50. ResNet50 Layer1-4 use all the features output by the middle layer of ResNet50. The major difference between these networks is the size of the output features. As shown in Table 1, the features output by Custom UNet have large width and height, while the output of ResNet50 Layer4 has small width and height and large number of channels.

4.2 Metrics

We prepared four evaluation metrics. The first is IoU, the quotient of union and intersection between the reconstructed mesh and the ground truth mesh. The second one is Chamfer L1 distance, which is calculated as the average distance from the point on the reconstructed mesh to the nearest neighbor point on the ground truth mesh and the distance from the point on the ground truth mesh to the nearest neighbor point on the reconstructed mesh. The third is the food volume error, which represents the absolute volume error of the food. The food volume is calculated from the difference between the dish and the plate. The fourth is the relative food volume error, which is the relative volume error of the food. This metric is the ratio of how much the inferred food volume differs from the ground truth food volume.

Table 1: Output feature shape for each encoder input feature shape.

backbone	input	output
Custom UNet	$3 \times 224 \times 224$	$128 \times 112 \times 112$
ResNet50 (Layer 1)	$3 \times 224 \times 224$	$255 \times 56 \times 56$
ResNet50 (Layer 2)	$3 \times 224 \times 224$	$512 \times 28 \times 28$
ResNet50 (Layer 3)	$3 \times 224 \times 224$	$1024 \times 14 \times 14$
ResNet50 (Layer 4)	$3 \times 224 \times 224$	$2048 \times 7 \times 7$

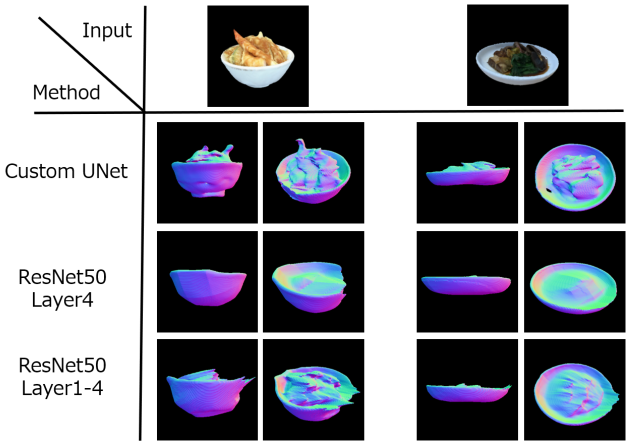

Figure 5: Comparison of reconstruction results with different encoders.

4.3 3D reconstruction of dish and plate

First, we made experiments using the three types of encoders mentioned above to see if our method could reconstruct 3D shape correctly. Image dataset A was relatively easy to learn and was used for training. The results of the quantitative evaluation are shown in Table 2. As a result, the method using Custom Unet was superior in both reconstruction accuracy and volume estimation.

Next, as for the qualitative evaluation, the reconstruction results are shown in Figure 5. It can be seen that the case using Custom UNet was the most accurate quantitatively, but it was also good qualitatively. It can be seen that the final output of ResNet50 completely loses the details of the food shape, and even when the intermediate features of ResNet50 are used, the details of the meal are not captured correctly compared to the method using Custom Unet. From this experiment, it is clear that the size of encoder features is extremely important.

Table 2: Quantitative results of 3D reconstruction of dish and plate using RGB-D images. In the table, C-L1 means Chamfer L1 distance, FVE means Food Volume Error, and r-FVE means relative Food Volume Error.

encoder	C-L1 (dish)	IoU (dish)	C-L1 (plate)	IoU (plate)	Mean FVE (cm^3)	Median FVE (cm^3)	Mean r-FVE	Median r-FVE
Custom UNet	**0.00341**	**0.702**	0.00581	**0.537**	**73.253**	**46.046**	0.595	**0.13**
ResNet50 (Layer 4)	0.00445	0.636	0.00607	0.437	167.271	99.129	0.658	0.377
ResNet50 (Layer 1-4)	0.00516	0.558	**0.00566**	0.470	80.4859	54.293	0.633	0.166

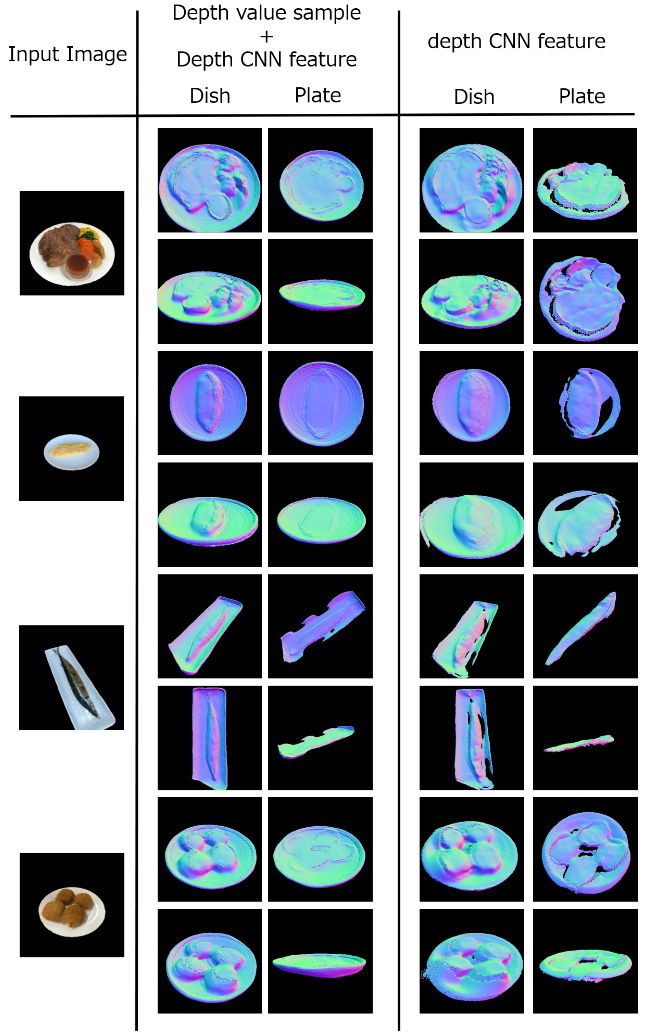

Figure 6: Comparison of models using depth-value sampling and depth features with models using only depth features. It can be seen that the depth value sampling contributes to the clear contours of the dish reconstruction results and the accurate reconstruction of the plate.

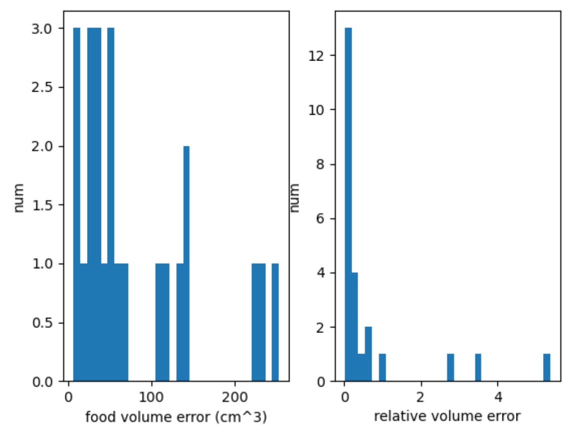

Figure 7: Distributions of volume errors were calculated for the evaluated dataset using the method with the highest accuracy. The left panel shows the distribution of absolute volume error, and the right panel shows the distribution of relative volume error. The mean of absolute volume error is 79.524 (cm^3), and the median is 51.241 (cm^3). The mean and median of the relative errors are 0.688 and 0.15, respectively.

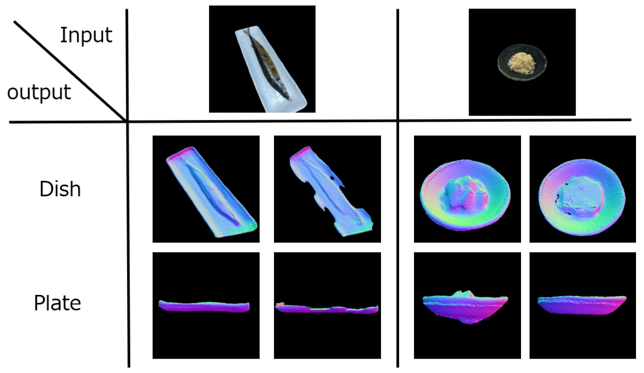

Figure 8: The two input images and reconstruction results that yielded the worst relative errors.

4.4 The effect of depth image utilization methods.

When the proposed method is used in an application, the depth is not always fixed, as in the case of Image dataset A. Therefore, we used Image dataset B, which has a more realistic setting for training, to experiment with how handling depth images contributes to accuracy. We used the Custom UNet, the most accurate encoder for all the methods. The results are shown in Table 3. Table 3 shows that the method using both depth value sampling and depth features is the most accurate. Figure 6 shows qualitatively how the reconstruction results differ between the best method and the method using only depth features. It can be seen that not only the depth

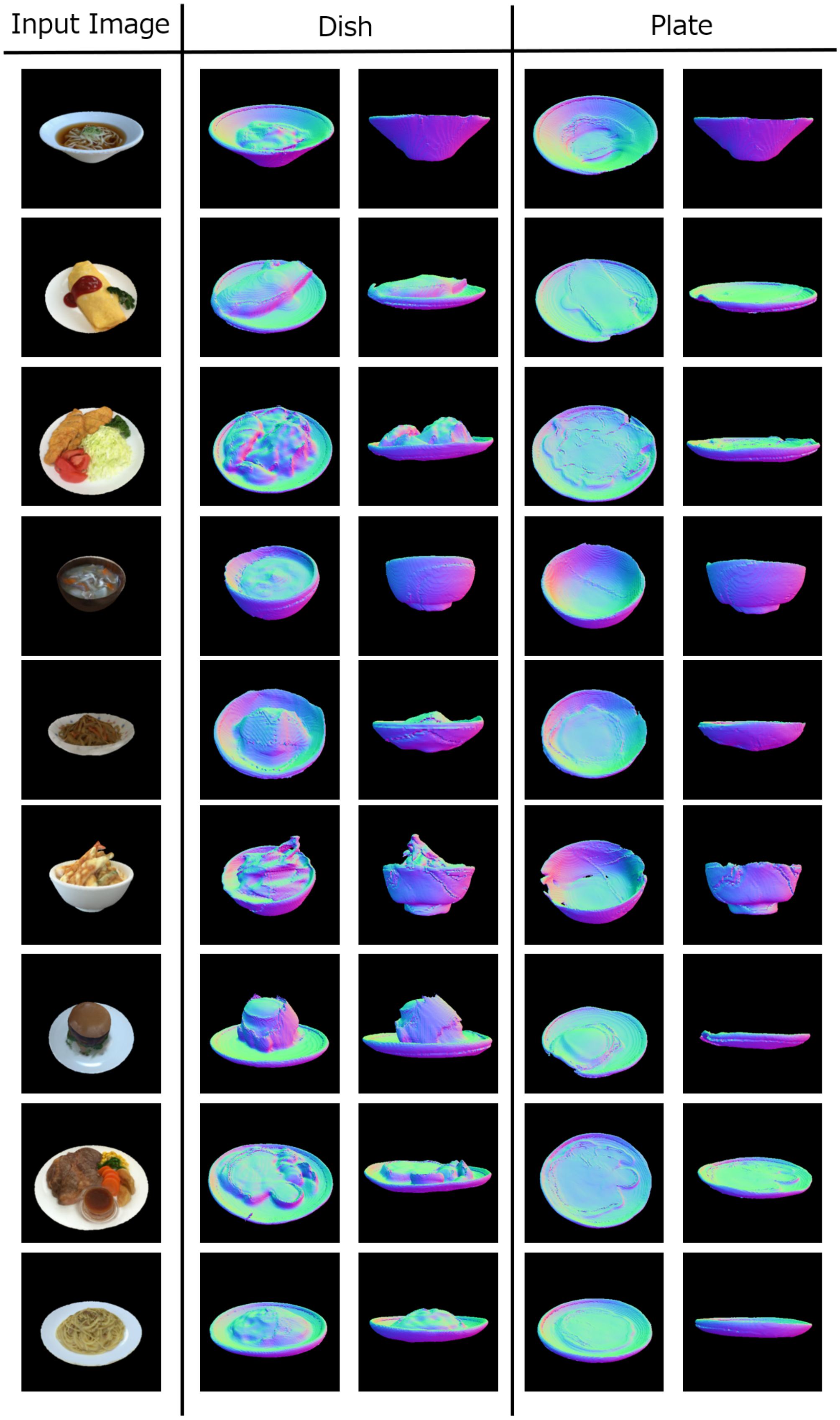

Figure 9: The results are trained on Image dataset B using Custom UNet as encoder and the method used for depth value sampling and depth features. Inputs are an RGB-D image and camera parameters.

Table 3: Comparison of which method of utilizing depth images gives the best accuracy. In the table, S means depth value sampling, and C means depth features obtained from CNN. Otherwise, the notation is the same as in Table 2.

Depth	valid	C-L1 (dish)	IoU (dish)	C-L1 (plate)	IoU (plate)	Mean FVE (cm^3)	Median FVE (cm^3)	Mean r-FVE	Median r-FVE
C, S	24/24	**0.00307**	**0.567**	**0.00498**	**0.407**	**79.524**	**51.24**	0.688	**0.150**
C	24/24	0.00333	0.534	0.00592	0.337	112.314	90.291	**0.616**	0.259
S	23/24	0.00847	0.356	0.0104	0.124	125.344	104.122	0.791	0.32
None	1/24	invalid	invalid	invalid	invalid	invalid	invalid	invalid	invalid

image feature but also the depth value sampling is used as the input of the decoder, resulting in clear contours of the reconstructed dish and accurate reconstruction of the plate. Figure 9 shows the reconstruction results of the most accurate method using Custom UNet as an encoder, depth value sampling and depth features.

4.4.1 Volume Error Analysis. We analyzed the volumetric error of the model with Custom UNet and depth value sampling and depth features, which was the most accurate model in Table 3. The distribution of the volume error is shown in Figure 7. The absolute error is less than 50cm^3, and the relative error is less than 0.2 for most of the data, indicating good accuracy. Two inputs with extremely poor relative errors and their reconstruction results are shown in Figure 8. In both cases, the reconstructed tableware is thinner or chipped compared to the meal. The plate consistency loss is a loss function that prevents a situation in which the dish does not occupy the space occupied by the plate, but the opposite is not correctable, which may be the reason for this problem.

5 CONCLUSION

In this paper, we propose “Real Scale Hungry Networks”, a real-size 3D reconstruction method using RGB-D images and camera models to solve the problem of our previous work, “Hungry Networks” [12], that the reconstructed objects are not real scale. By utilizing depth images, this method reconstructs 3D objects accurately as they are in real scale without normalizing 3D shapes. Also, using the reconstruction results, a highly accurate estimation of actual volume is realized. In future work, we would like to utilize the estimated volume for calorie estimation and other purposes.

Acknowledgment This work was supported by JSPS KAKENHI Grant Numbers, 21H05812, 22H00540, 22H00548, and 22K19808.

REFERENCES

[1] Y. Ando, T. Ege, J. Cho, and K. Yanai. 2019. DepthCalorieCam: A Mobile Application for Volume-Based FoodCalorie Estimation using Depth Cameras. In *Proc. of the 5th International Workshop on Multimedia Assisted Dietary Management.* 76–81.

[2] M. Y. Chen, Y. H. Yang, C. J. Ho, S. H. Wang, S. M. Liu, E. Chang, C. H. Yeh, and M. Ouhyoung. 2012. Automatic chinese food identification and quantity estimation. In *Proc. of SIGGRAPH Asia 2012 Technical Briefs.* 1–4.

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

[4] T. Ege and K. Yanai. 2017. Imag-Based Food Calorie Estimation Using Knowledge on Food Categories, Ingredients and Cooking Directions. In *Proc. of ACM Multimedia Thematic Workshop.*

[5] T. Ege and K Yanai. 2018. Image-Based Food Calorie Estimation Using Recipe Information. *IEICE Transactions on Information and Systems* E101-D, 5 (2018), 1333–1341.

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

[7] F. Kong and J. Tan. 2011. DietCam: Regular Shape Food Recognition with a Camera Phone. In *2011 International Conference on Body Sensor Networks.* 127–132.

[8] j. Li, Y. Liu, Gong D., Q. Shi, X. Yuan, C. Zhao, and I. Reid. 2019. RGBD Based Dimensional Decomposition Residual Network for 3D Semantic Scene Completion. In *Proc. of IEEE Computer Vision and Pattern Recognition.*

[9] Y. Lu, D. Allegra, M. Anthimopoulos, F. Stanco, G. M. Farinella, and S. Mougiakakou. 2018. A multi-task learning approach for meal assessment. In *Proc. of the Joint Workshop on Multimedia for Cooking and Eating Activities and Multimedia Assisted Dietary Management.* 46–52.

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

[11] A. Meyers, N. Johnston, V. Rathod, A. Korattikara, A. Gorban, N. Silberman, S. Guadarrama, G. Papandreou, J. Huang, and K. P. Murphy. 2015. Im2Calories: towards an automated mobile vision food diary. In *Proc. of the IEEE International Conference on Computer Vision.* 1233–1241.

[12] S. Naritomi and K. Yanai. 2020. Hungry Networks: 3D Mesh Reconstruction of a Dish and a Plate from a Single Dish Image for Estimating Food Volume. In *Proc. of ACM Multimedia Asia.*

[13] Y. Nie, X. Han, S. Guo, Y. Zheng, J. Chang, and J. J. Zhang. 2020. Total3DUnderstanding: Joint Layout, Object Pose and Mesh Reconstruction for Indoor Scenes From a Single Image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[14] M. Puri, Zhiwei Zhu, Q. Yu, A. Divakaran, and H. Sawhney. 2009. Recognition and volume estimation of food intake using a mobile device. In *2009 Workshop on Applications of Computer Vision (WACV).* 1–8.

[15] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li. 2019. PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization. In *Proc. of IEEE International Conference on Computer Vision.*

[16] S. Song, F. Yu, A. Zeng, Angel X Chang, M. Savva, and T. Funkhouser. 2017. Semantic Scene Completion from a Single Depth Image. In *Proc. of IEEE Computer Vision and Pattern Recognition.*

[17] Q. Thames, A. Karpur, W. Norris, F. Xia, L. Panait, T. Weyand, and J. Sim. 2021. Nutrition5k: Towards Automatic Nutritional Understanding of Generic Food. In *Proc. of IEEE Computer Vision and Pattern Recognition.*

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

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