

Iterative Coarse-to-Fine 6D-Pose Estimation Using Back-propagation

Ryosuke Araki¹, Kohsuke Mano¹, Tadanori Hirano²,
Tsubasa Hirakawa¹, Takayoshi Yamashita¹, and Hironobu Fujiyoshi¹

Abstract—We propose a 6D pose estimation method for an object from a single RGB image for a robotic grasping task. Many approaches estimate pose parameters from images taken from other viewpoints and use deep learning to achieve high accuracy. However, most of these methods are not robust to changes in object texture, and there is a possibility that the correct pose cannot be estimated by only one-time inference. Our aims are to reduce the number of failure cases and improve the accuracy by a novel architecture using the iterative backpropagation of a pose decoder network and pose estimation on intermediate representation. The error between random and target pose parameters are backpropagated to a neural network and the gradient for approaching the target pose is obtained. The pose parameter is updated using the obtained gradient, the error is calculated again, and backpropagation is re-performed. Repeating this process, we estimate a more accurate pose. Experiments using our own dataset show that estimation accuracy is improved and the number of failure cases is reduced. Furthermore, estimation by coarse-to-fine iterative processing is more accurate and faster. We also experiment with grasping using a UR5 robot and show that the robot can grasp objects without depth information when using the pose estimated by the proposed method.

I. INTRODUCTION

The rapid progress of vision technology has contributed significantly to solving a number of robotics tasks, including the grasping detection of an object. Many methods have been proposed for grasping objects by robots. For example, detecting direct grasping from an RGB image captured by a camera and converting it to a world coordinate system [1], [2], [3], [4], and estimating an object’s pose and linking its graspable points [5], [6], [7]. If we wanted the robot to simply grasp the object, we would adopt the former method. However, robots that stock products in a grocery or convenience store not only grasp objects but also scan barcodes on products and check expiry or best-before dates. For example, the “Future Convenience Store Challenge (FCSC) [8]”, one of the competitions at the World Robot Summit held in Tokyo in 2018, had a “stocking and disposing task.” This task required checking the expiry date of a product on the display shelf, discarding expired products, and restocking non-expired products. In this case, as the robot would need to check the surface on which the barcode and expiry date is shown, a method that estimates the pose of the object is suitable.

¹ Machine Perception and Robotics Group, Chubu University, 1200 Matsumoto-cho Kasugai-shi Aichi, JP {ryorsk@mprg.cs, runrun70@mprg.cs, hirakawa@mprg.cs, yamashita@isc, fujiyoshi@isc}.chubu.ac.jp

² CKD Corporation, 250 Oji-2 Komaki-shi Aichi, JP t-hirano@ckd.co.jp

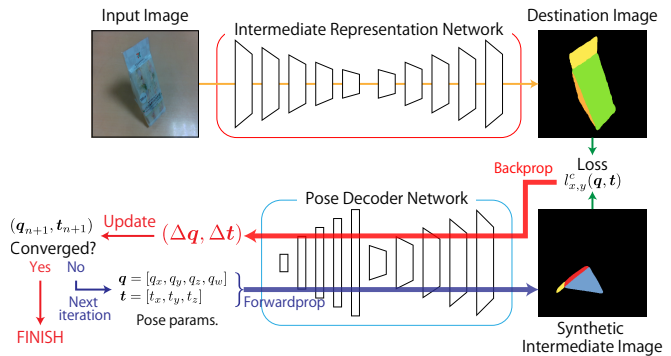


Fig. 1. **Pose estimation flow of our approach.** The detail of the coarse-to-fine iterative update (dashed blue and red arrows) is shown in Fig. 5. The model details of the pose decoder network are shown in Fig. 3.

The aim of an object pose estimation task is calculating a matrix \mathbf{R} representing the three-dimensional rotation of an object with the camera coordinates as the origin, and a translation \mathbf{t} . Many methods of estimating an object’s pose have been proposed, and these can be divided broadly into two types: model-based and appearance-based methods. A model-based method matches a 3D CAD model with a point cloud obtained from a sensor. For accurate pose estimation, this process is performed repeatedly. However, this does not work if the acquired point cloud has much noise or lacks a number of keypoints. On the other hand, an appearance-based method learns images captured from multiple viewpoints for the pose estimation. Thanks to the advancements of deep learning, there are many appearance-based methods, but changes in the texture of an object and lighting affect the pose estimation. Also, since it is not possible to perform iterative processing like model-based approaches, the estimated pose is not very reliable.

We propose a 6D pose estimation method by iterative processing that introduces intermediate representation for pose estimation. Because this method uses only RGB images as input, it is robust against a lack of point clouds and changes in appearance, which simplifies the system. The first step of pose estimation is replacing the target object image with an intermediate representation image. It absorbs changes in texture and lighting, so changes in appearance and texture do not affect pose estimation (see Sec. III-A). Next, random values are given to the well-trained pose decoder network as initial pose parameters. We calculate the error between the intermediate representation image generated by inputting these initial pose parameters and that of the target

object, and this error is backpropagated to the pose decoder network (see Sec. III-B). In this way, we acquire the update amount value of the random values and the pose parameters of the target object (i.e., the desired pose parameters). After that, each pose parameter is updated using this value. By repeating this process, it finally estimates accurate pose parameters. Furthermore, in the initial stage of the matching, iterative processing is performed with a coarse-resolution image, and when obtaining a certain accuracy, the iterative processing is performed with a high-resolution image. This coarse-to-fine processing improves the estimation accuracy. This method is based on an object’s appearance but does not use multi-view images, so it is a challenging task.

The results of an evaluation using our dataset show that our method achieves a higher performance than conventional methods. We also show that coarse-to-fine processing estimates a pose with fewer updates.

A brief summary of our contributions is as follows:

- Our proposed approach estimates a pose more accurately by repeatedly backpropagating the estimation error to a generator network. This approach is an entirely novel idea that was not found in conventional pose estimation methods.
- The intermediate representation absorbs changes in texture and lighting, so changes in appearance and texture do not affect pose estimation.
- During pose estimation, the network performs coarse-to-fine processing to avoid slowdowns.
- Our approach has the potential to be adapted to various objects by changing and combining intermediate expressions.
- Robotic grasping experiments show that the robot can grasp the objects without depth information.

II. RELATED WORK

We introduce the methods of object pose estimation that have been proposed, and they are roughly classified into two types: model-based and appearance-based.

A. Model-based Pose Estimation

Before the appearance of deep learning, many model-based pose estimation methods were proposed. These methods basically match a 3D CAD model with a point cloud obtained from a sensor. Iterative Closest Point (ICP) [9] is a well-known method, and many derived algorithms have been proposed. The estimation flow of ICP is the corresponding point search, pose estimation, and registration. After searching for multiple corresponding points from two point clouds to be registered, a rotation matrix \mathbf{R} and a translation matrix \mathbf{t} are estimated from the corresponding points. Then, the registration is performed using the estimated parameters. This action is repeated multiple times to increase the estimate accuracy. To reduce the effect of backgrounds and disturbance, SegICP [10] detects the target object using semantic segmentation and performs ICP. Drost *et al.* [11] match a pair of points with normal information and features. Brachmann *et al.* [12] estimate probability maps for objects

to be detected and obtain robustness against illumination changes by using a random forest. Hodañ *et al.* [13] use detection and hashing to efficiently extract point clouds of objects. Manhardt *et al.* [14] proposed a method of iterative refinement using a 3D CAD model instead of a point cloud.

The problem with a model-based method is that if the acquired point cloud has noise or is lacking sufficient points, it cannot search the corresponding points nor estimate pose. Therefore, it is difficult to apply it to translucent objects, ones with specular reflection, and black ones.

B. Appearance-based Pose Estimation

Since the appearance of the convolutional neural network (CNN), the number of appearance-based pose estimation methods using deep learning have increased. Most of them estimate poses by learning images taken from multiple viewpoints and point cloud.

Render for CNNs [15] is a method to directly estimate the object class and camera viewpoint for a single image using a CNN. Multi-view multi-class CNN [16] uses multiple RGB-D images as input for pose estimation. SSD-6D [7] estimates 6D poses using a model based on Single Shot Multibox Detector (SSD) [17], a breakthrough algorithm for object detection. SSD-6D estimates the class and pose of the object at the same time from a single RGB image. The pose is estimated using the viewpoint and in-plane angle, and it is a classification task in increments of 5 degrees. Single Shot 6D [18] estimates 6D poses using a model based on You Only Look Once (YOLO) [19], which is another breakthrough in object detection like SSD. BB8 [20] restricts the range of the pose in advance and estimates pose using an RGB image. PVNet [21] estimates pose in two stages: feature point extraction and Perspective-n-Point. Normals for each pixel in relation to the feature points are calculated, and RANSAC-based voting is performed to avoid uncertain feature points. As a result, it estimates pose in consideration of uncertainty. The outputs of the network are the semantic labels of the objects and the normals to feature points. Sundermeyer *et al.* [22] proposed the Augmented AutoEncoder (AAE). They extract a target object using AAE and estimate the pose of the object by mapping the pose parameters to the latent vectors of AAE.

Since appearance-based methods do not use point clouds, the model-based method is robust against objects that are difficult to estimate. However, changing the texture and light source position of the object affects the pose estimation. In other words, the features extracted from the image are insufficient for pose estimation. Thus, both model-based and appearance-based methods have their advantages and disadvantages. It is desirable to perform pose estimation repeatedly, like ICP, as an alternative idea. However, it is not realistic to repeat the process many times using a generation model that has a high computational cost.

III. PROPOSED METHOD

We propose a novel 6D pose estimation method using iterative processing and an intermediate representation that is ro-

bust against missing point clouds and changes in appearance. The main feature of this method is parameter estimation by backpropagation using a simple and differentiable generator network. Fig. 1 shows the flow of pose estimation by our method.

First, an arbitrary object detection method detects an object to be estimated among multiple objects. The detected object image is converted into an intermediate representation called a *destination image* using a pre-trained intermediate representation network (orange arrow in Fig. 1). Then, the randomly determined initial pose parameters (\mathbf{q}, \mathbf{t}) are input to the pre-trained pose decoder network and generate an intermediate representation image of the pose parameters called a *synthetic intermediate image* (blue arrows in Fig. 1). The error between the destination image and the synthetic intermediate image (green arrows in Fig. 1) is backpropagated to the pose decoder network, and the gradient of the pose parameter is calculated (red arrows in Fig. 1). Thus, the network outputs the difference between the random parameters and the pose parameters of the target object (i.e., the desired pose parameters). When this process is repeated and certain parameters are obtained, the pose decoder network outputs a higher resolution synthetic intermediate image, and the process is repeated again (dashed blue and red arrows in Fig. 1). It finally estimates accurate pose parameters by this *coarse-to-fine* processing.

This method can estimate by iterative processing without a point cloud. In other words, there is no need to perform any refinement using ICP after pose estimation as performed in the conventional method. Pose estimation is completed only by this process.

A. Intermediate Representation Network

We introduce intermediate representations for two reasons. The first reason is the acquisition of robustness against changing textures and light sources as in conventional methods. The intermediate representation absorbs changes in texture and lighting, so changes in appearance and texture do not affect pose estimation. In addition, for a rectangular parallelepiped or cubic object, our method reduces the number of cases where the object is inverted during estimation by considering each surface in the intermediate representation. The second is to update the pose parameters by backpropagation by associating the object’s pose parameters and the pose image with the differentiable network. We do not need to focus on a specific intermediate representation, and we can select an appropriate intermediate representation depending on the object or scene. We adopt surface segmentation as intermediate expressions.

We use SegNet [23], which is a semantic segmentation network for the surface segmentation. This model learns each surface and part instead of the object class as a surface segmentation network. We assigned labels to each surface in principle as shown in Fig. 2 (a), but assigned a separate class for parts that are difficult to assign to other planes, such as the flap part (yellow). In this sandwich example, the model recognizes seven classes, including the background.

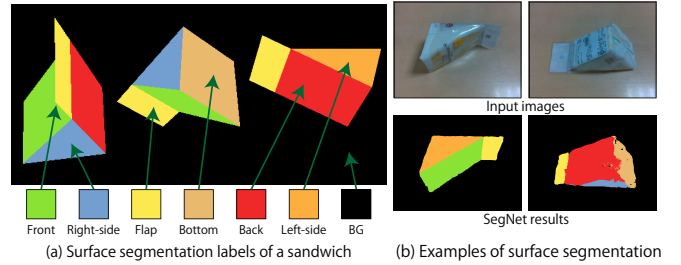


Fig. 2. **Surface segmentation labels and SegNet results of sandwich.** (a) The sandwich consists of several surfaces and flaps and is divided into seven classes. (b) Inference results when learning surface segmentation of sandwiches on SegNet.

We perform the same data augmentation as with the object segmentation.

Fig. 2 (b) shows the result of learning SegNet by using 1,117 pairs of real images and ground truth labels of surface segmentation. Although there is slight misidentification, the model recognizes all surfaces. As described later, small misrecognitions or lack of recognition does not harm the estimation.

B. Pose Estimation by Iterative Update using Pose Decoder

The pose decoder network takes the rotation parameters $\mathbf{q} = [q_x, q_y, q_z, q_w]$ and translation parameters $\mathbf{t} = [t_x, t_y, t_z]$, and generates the intermediate representation image $G(\mathbf{q}, \mathbf{t})$ for pose estimation. We call a generated image a “*synthetic intermediate image*”. When a rotation and translation vector is input, the network generates a corresponding intermediate representation image. To reduce the number of parameter values and avoid gimbal locks, we use quaternions for the rotation parameter. We calculate the loss between the destination image and synthetic intermediate image to perform the iterative process, and it is backpropagated to the pose decoder network. The loss at each pixel (x, y) is calculated by

$$l_{x,y}^c(\mathbf{q}, \mathbf{t}) = (T_{x,y} - G(\mathbf{q}, \mathbf{t})_{x,y}^c)^2, \quad (1)$$

where c is each class and $T_{x,y}$ is the destination image. In Eq. (1), the class that is estimated to be the correct for each pixel is checked. If both the label and the estimated class are backgrounds, the loss is calculated to be 0; otherwise, the loss is calculated so that the probability of the correct class approaches 1. Note that the loss is calculated for each pixel, divided by the number of valid pixels, and averaged. Even if the synthetic intermediate image causes partial misrecognition, it has no adverse effect if the number of correctly identified pixels is much larger.

Next, the obtained loss is backpropagated to the pose decoder network to calculate the difference Δq_i and Δt_i :

$$\begin{aligned} \Delta q_i &= \sum_x^X \sum_y^Y \sum_c^C \frac{\partial l_{x,y}^c(\mathbf{q}, \mathbf{t})}{\partial q_i} \\ \Delta t_i &= \sum_x^X \sum_y^Y \sum_c^C \frac{\partial l_{x,y}^c(\mathbf{q}, \mathbf{t})}{\partial t_i} \end{aligned} \quad (2)$$

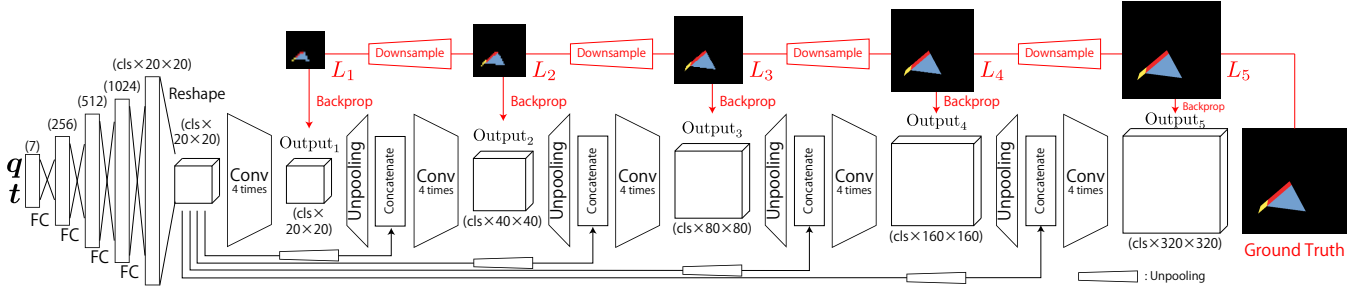


Fig. 3. **Model details of pose decoder network.** This is an example of a network that generates a surface segmentation image.

TABLE I

UPDATE RATE AT EACH RESOLUTION.

Resolution [pix]	20	40	80, 160, 320
$l_{x,y}^c(\mathbf{q}, \mathbf{t})$			≥ 0.2 < 0.2
η_q	0.01	0.01	0.03 0.005
η_t	0.0001	0.0001	0.0005 0.0001

Then, we update the pose parameters. The update formula is

$$\begin{cases} \mathbf{q}_{n+1} \leftarrow \mathbf{q}_n - \eta_q \Delta \mathbf{q}_n \\ \mathbf{t}_{n+1} \leftarrow \mathbf{t}_n - \eta_t \Delta \mathbf{t}_n \end{cases} \quad (3)$$

where $\eta_{\{q,t\}}$ is the update rate, which depends on the resolution of the image and value of Eq. (1) as shown in Table I. This parameter is input to the pose decoder network again, and the same process is repeated. If the value of Eq. (1) is smaller than the threshold, the iterative process ends. This threshold depends on the resolution of the image and is described in detail in the next section.

We use the random parameters as the initial pose parameters for pose estimation, but these parameters often cannot be updated properly when the overlap between the synthetic intermediate image and the destination image is small. Therefore, we take 20 patterns of random parameters, calculate Eq. (1) for each parameter, and use the parameters with the smallest value of Eq. (1) as the initial pose.

C. Details of Pose Decoder Network

We were inspired to create this original model by the generative adversarial networks (GANs) [24] by Goodfellow *et al.* and their derivatives: deep convolutional GANs (DCGANs) [25] and progressive growing of GANs (PGGANs) [26]. Fig. 3 shows the configuration of our pose decoder network. Like PGGANs, our network uses a strategy of increasing the resolution of low-resolution generated images. However, in our network, there is no discriminator and all generated images for each resolution are backpropagated simultaneously. When the network obtains the feature maps of $\{20 \times 20, 40 \times 40, 80 \times 80, 160 \times 160, \text{ and } 320 \times 320\}$, the loss is calculated between the ground truth label downsampled to each resolution and each generated image, and backpropagation is performed. As a result, the network outputs an intermediate representation necessary for achieving coarse-to-fine pose estimation. Furthermore, it generates features near the edge of the object more accurately. In

this structure, the number of discriminators increases, and optimization becomes difficult, so we only use the generator. In the Conv. block, four convolutions are performed, and the following parameters are set: filter size: 3×3 , stride: 1, and padding: 1. During the first three repetitions in each block, the output value is input to the activation function (ReLU). In addition, since the features of the pose parameters disappear when the unpooling is performed, the feature map of each resolution and the output of the first FC layer are concatenated.

Our network structure was adapted from that of the Synthesizer CNN proposed by Oberweger *et al.* [27], but ours is much deeper. Furthermore, the crucial difference is that Oberweger *et al.* estimate the pose with the Updater CNN, while we can estimate the pose by backpropagating the pose decoder network. This is a novel approach that not only reduces the number of CNNs to be trained, but also uses gradients from backpropagation for pose estimation.

1) *Loss Function:* The loss function of the pose decoder network is the average of loss at each resolution: $Loss = (L_1 + L_2 + L_3 + L_4 + L_5)/5$, where $L_{\{1,2,3,4,5\}}$ are the losses at each resolution ($\{20 \times 20, 40 \times 40, 80 \times 80, 160 \times 160, \text{ and } 320 \times 320\}$).

These losses depend on the intermediate representation. The loss of object/surface segmentation is calculated by the mean squared error for each class.

2) *Training Data and Method:* An intermediate representation image for training is created as follows: (1) a random pose parameter is determined, (2) the 3D CAD model of the object is rotated and translated (Fig. 4(a)) with the pose parameter, and, (3) an image is synthesized by perspective projection (Fig. 4(b)). The random pose parameter determined in (1) is used as the input, and the image rendered in (3) is used as the ground truth label.

The optimizer is the Momentum SGD with the following parameters set: learning rate: 0.0001 and momentum: 0.9.

D. Coarse-to-Fine Iterative Update

The resolution of the generated image is a trade-off relation between the accuracy of matching and the estimated speed. To avoid this trade-off, we perform the coarse-to-fine iterative update and achieve high-speed and high-precision estimation. First, matching is performed with a coarse-resolution image. When both the rotation and translation losses are lower than their respective thresholds, the process

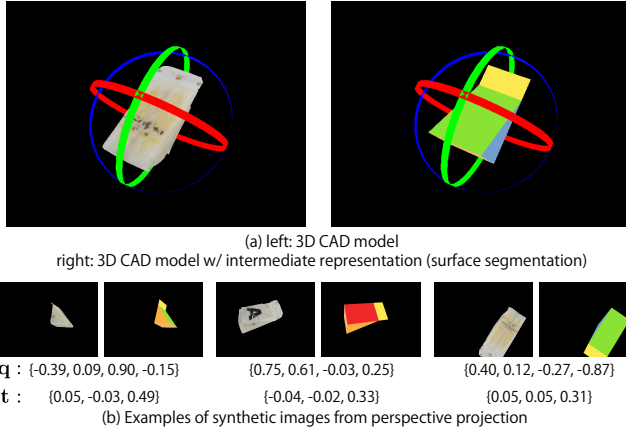


Fig. 4. **Example of the 3D CAD model and the images obtained by perspective projection transformation.** This example converts a sandwich to a surface segmentation. (a) The 3D CAD model of the intermediate representation is made by replacing the textures or creating a model with a similar shape. (b) We rotate and translate with random parameters and perform perspective projection transformation.

TABLE II
RELATIONSHIP BETWEEN RESOLUTION AND THRESHOLD.

Res. [pix]	20	40	80	160	320
Thresh.	0.001	0.001	0.00005	0.00005	0.00005

proceeds to the next step. At this time, if any parameter falls below its threshold, it is fixed and the other parameters continue to update. Next, using the parameters obtained in the previous process as initial pose parameters, the same process is performed on a higher resolution image. By repeating this process over multiple resolutions, the accuracy is higher than the estimation with a single resolution, which reduces the processing time. Table II shows the thresholds for each resolution. We define this threshold on the basis of the gradient when the image is changed by only several pixels.

IV. EXPERIMENT

We experimented with our pose estimation method using our own datasets. We also conducted ablation experiments to confirm the effect of the coarse-to-fine iterative update.

A. Dataset and Evaluation Metrics

1) *Dataset*: We used the Convenience Store Products (CSP) dataset created for 6D pose estimation of store products. Our approach assumes pose estimation for grasping detection used by a robot working in a convenience store. Therefore, we created a dataset for 6D pose estimation using four products sold at convenience stores in Japan. Target products are a sandwich, riceball, cup-a-soup, and lunchbox. As shown in Fig. 6, the included data is a 3D CAD model, surface segmentation information, and other intermediate representation data. These products are packaged in transparent plastic, making it difficult to obtain point clouds.

Using this dataset, we train two networks. The intermediate representation network uses RGB images of objects as

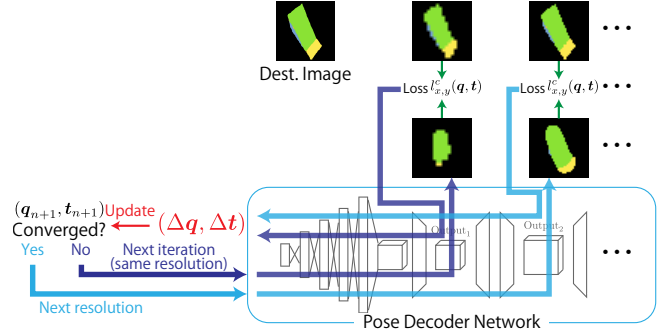


Fig. 5. **Flow of coarse-to-fine update.** First, estimation starts at a lower resolution. If both the rotation and translation losses are lower than their respective thresholds, the same process is performed on a higher resolution image.

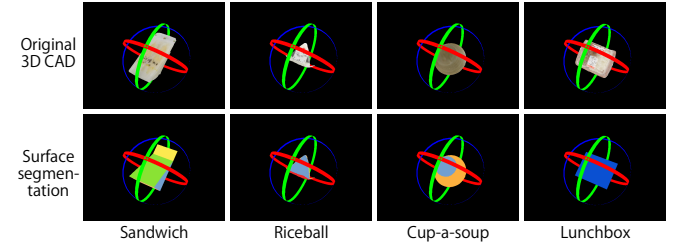


Fig. 6. **All items of the Convenience Store Products dataset.** The dataset contains 3D CAD models with real textures and the intermediate representation of each product.

input and intermediate representation images (e.g. surface segmentation) as ground truth labels. We train 200 sets of these pairs for each object. The pose decoder network is trained by preparing the data as mentioned in Sec. III-C.2. Since we can make countless pairs of input data and ground truth images, the network learns 4 million images (4 batchsize \times 1,000,000 iterations). The test data is a set of real RGB images and pose parameters, and 51 sets are prepared for each object.

2) *Evaluation Metrics*: We used the mean and standard deviation of the rotation and translation errors for each axis. The rotation error is the angle between the quaternion of the attitude parameter converted to the Euler angle and the correct angle for each rotation axis. The translation error is the difference between the translations of each axis.

B. Comparison with Other Methods

We experimented with the pose estimation of each object using CSP. We evaluated the performance using ICP and our proposed method. We prepared 51 pairs of ground truth poses and their intermediate representation images and compared the average and standard deviation of rotation error and translation error for each axis. As shown in Table III, the proposed method with surface segmentation achieved high accuracy. The accuracy is higher when using surface segmentation than when using object segmentation as an intermediate representation. Furthermore, our method achieved a smaller standard deviation than that of ICP, which means that the proposed method performs stable pose estimation. However,

TABLE III
COMPARISON USING CSP DATASET. THE VALUES IN PARENTHESES INDICATE THE STANDARD DEVIATION.

items	methods	Error of \mathbf{R} [deg]			Error of \mathbf{t} [mm]		
		R_x	R_y	R_z	t_x	t_y	t_z
sandwich	ICP	41.39 (38.02)	26.54 (18.23)	94.23 (51.86)	7.18 (15.31)	5.29 (7.07)	3.38 (1.85)
	ours	25.07 (20.89)	13.74 (17.43)	17.35 (31.83)	3.44 (3.74)	2.99 (2.41)	12.70 (9.77)
riceball	ICP	82.38 (43.81)	44.44 (28.29)	85.32 (56.24)	2.63 (1.57)	2.76 (1.51)	3.93 (2.77)
	ours	54.01 (35.00)	33.28 (28.14)	62.67 (64.22)	2.49 (1.63)	2.78 (1.58)	5.93 (5.88)
cup-a-soup	ICP	54.85 (51.04)	53.44 (38.14)	84.57 (56.51)	5.31 (10.31)	5.22 (6.55)	6.18 (6.41)
	ours	25.75 (38.99)	73.62 (36.26)	20.88 (50.21)	2.31 (2.22)	3.14 (3.33)	9.27 (17.36)
lunchbox	ICP	52.10 (41.60)	46.10 (30.73)	82.07 (55.81)	2.62 (1.47)	2.38 (1.32)	4.00 (2.96)
	ours	53.15 (48.19)	46.70 (31.55)	80.42 (63.67)	6.48 (5.54)	6.11 (5.22)	21.64 (14.4)

TABLE IV
COMPARISON FOR EACH RESOLUTION. "SINGLE": USING ONLY A SINGLE RESOLUTION, "C2F": USING THE COARSE-TO-FINE ITERATIVE UPDATES. THE VALUES IN PARENTHESES INDICATE THE STANDARD DEVIATION.

iteration type	resolution [pix]	Error of \mathbf{R} [deg]			Error of \mathbf{t} [mm]			#itr.	time [sec]
		R_x	R_y	R_z	t_x	t_y	t_z		
single	20	26.68 (23.90)	14.61 (17.61)	22.60 (37.04)	5.50 (5.74)	4.11 (3.73)	11.01 (11.36)	61	4.66
	40	25.61 (22.31)	13.90 (17.85)	19.10 (35.66)	4.17 (4.44)	3.32 (2.56)	11.22 (10.19)	63	4.98
	80	22.94 (22.19)	12.86 (17.87)	16.86 (35.71)	3.87 (4.02)	3.15 (2.57)	11.4 (10.37)	48	4.05
	160	23.06 (22.23)	12.61 (17.53)	16.71 (35.42)	3.83 (4.06)	2.98 (2.45)	11.49 (10.82)	47	5.38
	320	22.85 (22.32)	12.93 (17.71)	16.43 (34.65)	3.99 (4.08)	2.95 (2.44)	11.51 (11.01)	51	9.97
c2f	20 to 320	21.46 (22.15)	12.88 (17.90)	16.02 (35.22)	3.79 (3.77)	2.8 (2.44)	10.53 (10.82)	73	8.22

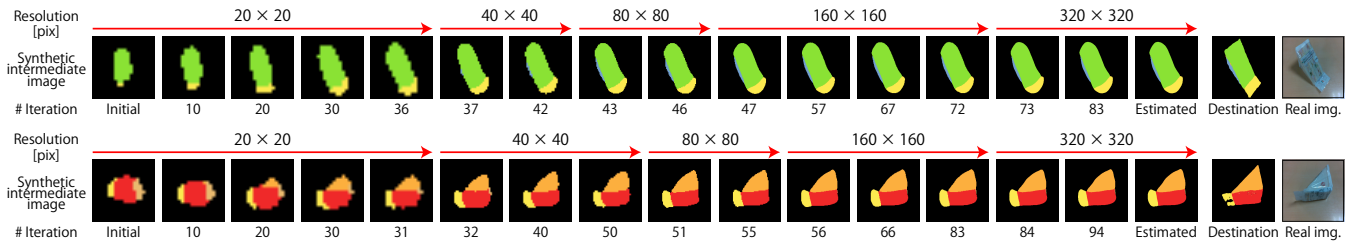


Fig. 7. Transition of pose estimation on sandwich.

























the error of t_z is high because our approach does not use the depth image. To solve this problem, it is necessary to use the other intermediate representation (e.g. a surface normal map) to cover this performance and to improve the accuracy of the pose decoder network.

C. Ablation Study for Coarse-to-Fine Iterative Update

We compared iterative processing at a single resolution and coarse-to-fine iterative update processing using multiple resolutions to confirm the usefulness of coarse-to-fine processing. We used the sandwich of CSP as experimental data. Table IV shows the average and standard deviation

of the number of updates of each pose parameter under each condition. When using a single resolution, 320 [pix] achieves the best accuracy. However, the speed is twice that of 20 [pix]. On the other hand, coarse-to-fine iterative update achieves the fastest estimation speed and improves accuracy. The speed of ICP is about 1 second, so it is inferior in terms of speed. The coarse-to-fine iterative update contributes to both high accuracy and high speed. Figs. 7 and 8 show the transition of pose estimation by the coarse-to-fine iterative update and the loss on sandwich, respectively. We also show these in the attached video. First, pose estimation

TABLE V
EXPERIMENTAL RESULTS OF ROBOTIC GRASPING.

Input images								
Segmentation results								
Pose estim. results								
1st grasp success	✓	✓	✓	✓	✓	×	×	×
Re-grasp success	-	-	-	✓	-			
Placement success	✓	✓	✓	✓	✓			

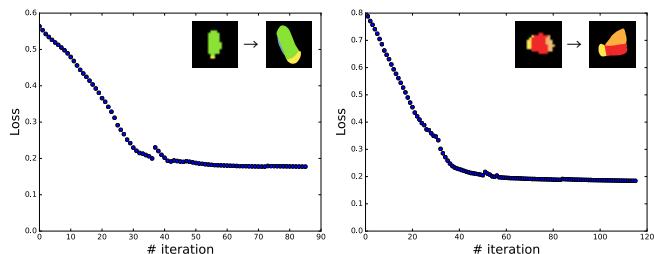


Fig. 8. Loss between synthetic intermediate image and destination image.

is performed at a low resolution, and the resolution is then gradually increased, and the estimated pose approaches the target pose. The graph (Fig. 8) shows that the loss between the synthetic intermediate image and the destination image has converged. When we perform the iterative update at a low resolution, the loss is stagnant, but we can resolve this issue by increasing the resolution.

D. Robotic Grasping Experiment

Our final goal is for a robot designed to operate in a convenience store to understand the correct pose of a product, grasp it, and dispose or display it again. To confirm that the proposed method can perform this task on the basis of the estimation results, we conducted an experiment using a UR5e arm robot, as shown in Fig. 9.

Although it is equipped with an Intel RealSense D435i vision sensor, which can be used to retrieve depth information, only the RGB camera was used. The task is to display a convenience store product (in this case, a sandwich) on a shelf in the correct position. Since our method assumes that the approximate position of an object detected using object detection, the sandwich is first placed in the picking area in a random position. We estimate the object’s pose and approach the object in the direction normal to the surface with the widest visible area to perform suction grasping. If the front of the object cannot be grasped, it cannot be displayed in the correct pose. Therefore, the robot places the object once on the picking area and re-grasps it from its front side.

Table V shows the results of the grasping experiment. The upper part of the table shows the input images, segmentation results, and pose estimation results for the eight patterns. The

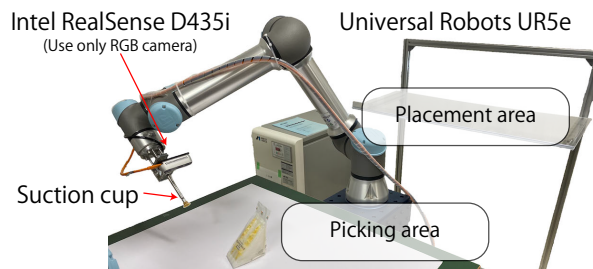


Fig. 9. Our robot system.

lower part shows the success and failure of first grasping the object (1st grasp), re-grasping the object (Re-grasp), and displaying the object in the placement area (Placement) using the pose estimation results. In the third case, the sandwich is of a different type, but the difference in texture is absorbed by segmentation, and the pose estimation and placement are successful. In the fourth case, since the object could not be grasped in front, we grasped the back of the object and displayed it on the shelf. The attached video shows the grasping process in the fourth case. There were a number of cases in which the estimation results strongly differed from the segmentation results on the basis of the pose, and there were those in which grasping was not possible even if the estimation results were approximately correct.

V. LIMITATION AND DISCUSSION

The low R_y of “cup-a-soup” indicates that it is difficult to estimate an object’s rotation on a cylinder. On the other hand, objects with easy-to-surface segmentation, such as “sandwich” and “rice ball” have relatively good accuracy. Since the accuracy depends largely on the choice of intermediate representations, we believe it is effective to increase the repertoire of intermediate representations and increase the number of objects for which pose estimation is possible. In addition, because we compute the error with a surface that is visible as an image, it may lead to the wrong estimation such as being flipped upside down. Therefore, we need to define a loss function that considers the positional surface relation. These are included in our future work.

The major limitation of our method is that the shape of the object must be simple enough to be estimated. If the object

is too complex to be represented by surface segmentation, the pose estimation becomes difficult. Therefore, when the proposed method estimates the poses of objects with complex shapes such as objects from the LineMOD and YCB datasets, it is effective to set the surface appropriately or use other intermediate representations such as a normal map.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a 6D pose estimation method using iterative processing that introduced an intermediate representation. The errors between random and target pose parameters are backpropagated to a pose decoder network and a gradient is obtained for approaching the target pose. The pose parameter is updated using the obtained gradient, the error is calculated again, and backpropagation is re-performed. By repeating this process, we can estimate a more accurate pose. Furthermore, estimation by coarse-to-fine iterative update is much faster.

We conducted experiments using the CSP dataset, and the results showed that the proposed method estimates the object pose with high accuracy. An ablation study showed that the coarse-to-fine iterative update estimated the object's pose with high accuracy while suppressing the speed reduction. Furthermore, grasping experiments using a robot showed that the estimated pose by the proposed method is accurate enough for grasping an object using a suction cup.

In this research, we attempted to pose estimation by repeatedly processing one image. However, assuming a real robot application, it is expected that we need to estimate the object's pose while moving a robot arm and plan an ideal moving-path. In the future, we will apply it to the path planning of a robot arm.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number JP20J15720.

REFERENCES

- [1] J. Glover, D. Rus, and N. Roy, "Probabilistic Models of Object Geometry for Grasp Planning," *Robotics: Science and Systems*, pp. 278–285, 2008.
- [2] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic Grasping of Novel Objects using Vision," *International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.
- [3] I. Lenz, H. Lee, and A. Saxena, "Deep Learning for Detecting Robotic Grasps," *International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.
- [4] J. Redmon and A. Angelova, "Real-Time Grasp Detection Using Convolutional Neural Networks," in *IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 1316–1322.
- [5] M. Nieuwenhuisen, D. Droschel, D. Holz, J. Stückler, A. Berner, J. Li, R. Klein, and S. Behnke, "Mobile Bin Picking with an Anthropomorphic Service Robot," in *IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2327–2334.
- [6] K. Harada, K. Nagata, T. Tsuji, N. Yamanobe, A. Nakamura, and Y. Kawai, "Probabilistic Approach for Object Bin Picking Approximated by Cylinders," in *IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3742–3747.
- [7] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making Rgb-Based 3D Detection and 6D Pose Estimation Great Again," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1521–1529.
- [8] K. Wada, "New Robot Technology Challenge for Convenience Store," in *IEEE/SICE International Symposium on System Integration*. IEEE, 2017, pp. 1086–1091.
- [9] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.
- [10] J. M. Wong, V. Kee, T. Le, S. Wagner, G.-L. Mariottini, A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, D. M. Johnson, et al., "SegICP: Integrated Deep Semantic Segmentation and Pose Estimation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2017, pp. 5784–5789.
- [11] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model Globally, Match Locally: Efficient and Robust 3D Object Recognition," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2010, pp. 998–1005.
- [12] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European Conference on Computer Vision*. Springer, 2014, pp. 536–551.
- [13] T. Hodaň, X. Zabulis, M. Lourakis, Š. Obdržálek, and J. Matas, "Detection and Fine 3D Pose Estimation of Texture-less Objects in RGB-D Images," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2015, pp. 4421–4428.
- [14] F. Manhardt, W. Kehl, N. Navab, and F. Tombari, "Deep Model-Based 6D Pose Refinement in RGB," in *European Conference on Computer Vision*, 2018, pp. 800–815.
- [15] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2686–2694.
- [16] C. Li, J. Bai, and G. D. Hager, "A Unified Framework for Multi-View Multi-Class Object Pose Estimation," in *European Conference on Computer Vision*, 2018, pp. 254–269.
- [17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot Multibox Detector," in *European Conference on Computer Vision*. Springer, 2016, pp. 21–37.
- [18] B. Tekin, S. N. Sinha, and P. Fua, "Real-Time Seamless Single Shot 6D Object Pose Prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.
- [19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.
- [20] M. Rad and V. Lepetit, "BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3828–3836.
- [21] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570.
- [22] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3D Orientation Learning for 6D Object Detection from RGB Images," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 699–715.
- [23] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.
- [25] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," in *International Conference on Learning Representations*, 2016.
- [26] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," in *International Conference on Learning Representations*, 2018.
- [27] M. Oberweger, P. Wohlhart, and V. Lepetit, "Training a Feedback Loop for Hand Pose Estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3316–3324.