SUMMARY   This paper proposes a method to create various training images for instance segmentation in a semi-supervised manner. In our proposed learning scheme, a few 3D CG models of target objects and a large number of images retrieved by keywords from the Internet are employed for initial model training and model update, respectively. Instance segmentation requires pixel-level annotations as well as object class labels in all training images. A possible solution to reduce a huge annotation cost is to use synthesized images as training images. While image synthesis using a 3D CG simulator can generate the annotations automatically, it is difficult to prepare a variety of 3D object models for the simulator. One more possible solution is semi-supervised learning. Semi-supervised learning such as self-training uses a small set of supervised data and a huge number of unsupervised data. The supervised images are given by the 3D CG simulator in our method. From the unsupervised images, we have to select only correctly-detected annotations. For selecting the correctly-detected annotations, we propose to quantify the reliability of each detected annotation based on its silhouette as well as its textures. Experimental results demonstrate that the proposed method can generate more various images for improving instance segmentation.

key words: instance segmentation, semi-supervised learning, 3D CG models, image synthesis

1. Introduction

Improvements in instance segmentation have been realized by the rapid progress of deep learning in recent years. As well as most current deep learning methods, instance segmentation as supervised learning requires a large amount of annotations. While instance segmentation requires the pixelwise region of each object and its class label as supervised data, it is difficult to annotate a lot of data. To solve this problem, we combine supervised learning using synthesized images [1] and semi-supervised learning [2] for automatic annotations.

In image synthesis, free-viewpoint training images can be synthesized by a 3D CG simulator. The variation of the synthesized images can be controlled in the CG simulator in terms of observation environments (e.g., illumination, camera parameters) if the 3D model of any object is provided. It is, however, difficult to synthesize a variety of object images because of difficulty in preparing a variety of 3D object models.

Semi-supervised learning uses a small set of supervised data and a large amount of unsupervised data for initial model training and model update, respectively. A number of unsupervised images can be collected from the Internet and useful for training the model, for example for object detection [3]-[5]. We utilize these images for our semi-supervised learning scheme which improves self-training.

Difficulty in applying self-training to instance segmentation is how to select only annotations that are correctly detected, as shown in Fig. 1 (a). To select the correctly-detected data, traditional self-training methods for object detection/segmentation measure only the class confidence of each estimation. However, since only a limited number of supervised data are used for pre-training the model of each class, the class confidence is sometimes unreliable so that another class has a high-class confidence, as shown in Fig. 1 (b). In addition, neither of overlaps of multiple instances nor small parts of an instance, as shown in Fig. 1 (c) and (d), are selected in our self-training strategy. This is because (1) multiple instances must be independently detected as different instances and (2) it is difficult to discriminate between different object classes in small regions. These problems make it difficult to employ semi-supervised learning for instance segmentation in still images.

Our method resolves these problems by combining semi-supervised learning with the aforementioned image synthesis.
synthesis approach. In addition to this first contribution of this work, we propose to strictly evaluate the silhouette of each instance detected in unsupervised images whether or not this instance can be used for semi-supervised learning. Finally, we made the foodstuff dataset for evaluating our method. Since several different foodstuffs have similar textures and silhouettes, its difficulty is sufficient for validating the effectiveness of the proposed method. These three contributions are proposed in this paper.

2. Related Work

2.1 Instance Segmentation

Instance segmentation needs to distinguish between different instances of the same class in addition to semantic segmentation such as FCN [6] and U-Net [7]. Hence, most of instance segmentation models are implemented by combining semantic segmentation and object detection (e.g., Faster-RCNN [8] SSD [9], and YOLO [10], [11]).

To combine object detection and segmentation, Dai et al. [12] used cascade network. Cascade network is composed of proposal-part, masking-part, and classification-part. Each part is conducted in order. The other way of combining object detection and segmentation is using multitask learning. FCIS [13] simultaneously detects segmentation and instance classes within a bounding box. In Mask-RCNN [14] proposed by He et al., segmentation and detection are jointly optimized. Additionally, RoI align is proposed to revise RoI pooling (method to reduce the feature map size to fixed size) by using subpixels for more accurate feature extraction.

These approaches use object detection to distinguish the instances. Therefore, it is difficult to distinguish the instances which are hard to separate by using the bounding box. To solve this problem, Kirillov et al. [15] used the boundary of the instances to distinguish them. On the other hand, Novotny et al. [16] used embedding method to identify the instances. If embedded vectors belong to the same instance, these vectors are embedded to be close.

These works explored better network architectures for instance segmentation in the condition that a sufficient amount of annotated images are given. While a large number of annotated images is difficult to be manually collected, this paper explores how to reduce the annotation cost.

2.2 Image Synthesis

A problem in using synthetic images is difference between synthetic and real-world images. This problem is addressed, for example, by domain adaptation and domain randomization.

Domain adaptation is used when the data distribution of training data differs from that of test data. Bousamlis et al. [17] transform the data distribution of the synthesized images to that of the real-world images through Generative Adversarial Network (GAN) [18]. While this method [17] achieves image-level adaptation using the generator of GAN, Ren et al. [19] use feature-level domain adaptation using the encoder of GAN. These methods are useful if it is easy to prepare real-world images corresponding to each synthesized image. However, it is not easy to use these domain adaptation methods when such real-world images are not available.

Even if such real-world images are unavailable, domain randomization [20] is applicable. In domain randomization, parameters for image synthesis (e.g., illumination, camera parameters) are randomized in a realistic range. This randomized image synthesis prevents over-fitting to a specific domain for updating a model so that it is applicable to a real-world domain. Tremblay et al. [1] successfully applied domain randomization to object detection.

While all aforementioned methods [1], [17], [19], [20] use a 3D CG simulator, it is difficult to prepare a variety of 3D object models. This problem is resolved in Cut, Paste and Learn [21] by pasting cropped images with different orientations and scales on various background images. How to collect a variety of such cropped images of objects is the focus of this paper.

2.3 Semi-Supervised Learning and Weakly-Supervised Learning

Since self-training [22] is simple as semi-supervised learning, it can be applied to a lot of problems. Rosenberg et al. [2] applied it to various object detection methods. The performance of self-training depends on whether or not only correct detections are used for model update. Co-training [22] is a straightforward way to mitigate this problem. In Co-Training, multiple models are used to determine whether or not each detection is correct.

While such semi-supervised learning approaches is successfully utilized for semantic segmentation [23], different instances must be separated for instance segmentation. As mentioned in Sect. 1, it is difficult to separate different instances in their overlap. Though separating such instances can be relatively easy in videos as demonstrated in [24], that is not easy in still images.

Weakly-supervised learning, which learns required annotations based on weak annotations, is another way to reduce the cost of data annotation. There are two types of weakly-supervised learning in instance segmentation. One of them employs bounding boxes each of which is labeled with its object class as weak annotations. Zhao et al. [25] use Graph Cuts [26] for extracting a pixelwise annotation in each given bounding box. The other type employs only class labels of objects observed in each image as weak annotations [27]–[29]. Based on local similarities in the images having the same class label, CNNs are trained for pixelwise instance segmentation.

3. Semi-Supervised Learning with Image Synthesis

Figure 2 shows an overview of our method consisting of the
following steps:

**Step 1** Pre-train the model by initial training dataset (Sect. 3.1)

**Step 2** Collect images for semi-supervised learning by weak labels (Sect. 3.2)

**Step 3** Select correct images after instance segmentation (Sect. 3.3)

**Step 4** Create additional training images (Sect. 3.4)

**Step 5** Update the model until convergence (Sect. 3.5)

### 3.1 Pre-Train the Model by Initial Training Dataset

We create annotated images required for initial learning by image synthesis using domain randomization [1], [20] utilizing a 3D CG simulator. The 3D models of target objects are created before this image synthesis step. In our experiments, the 3D model of each object was created by Recap [30] with its multi viewpoint images captured in our lab environment, as illustrated in Fig. 3. The 3D model is textured by the multi viewpoint images as shown in Fig. 3.

To avoid over-fitting to specific parameters in the 3D CG simulator, we randomize the following parameters: (1) the number and position of light sources, (2) the number, position, and orientation of target objects, (3) the position and orientation of a camera, (4) the number, position, orientation, texture, the 3D shape of distractor objects, and (5) the background images.

Distractor objects are used in [1] for improving domain randomization. Figure 4 shows examples of the distractors. The distractors allow us to reduce false-positive detections. In our experiments, the 3D shapes of the distractors were randomly selected from artificial 3D shapes (e.g., sphere, cylinder, cone, donut, square) and from those of the target objects. All images synthesized by the aforementioned process, Mask-RCNN [14], which is pre-trained by the COCO dataset [31], is finetuned. We call this training process 0-th training iteration.

### 3.2 Collect Images for Semi-Supervised Learning by Weak Labels

For semi-supervised learning, a large number of unsupervised training images are collected from the Internet. The unsupervised images are retrieved by keyword search. Examples of the retrieved images are shown in Fig. 6.

Each collected image associated with a retrieval keyword is used for updating the model of Mask-RCNN. If the retrieval process is always perfect (i.e., an object of the retrieval keyword is observed in the retrieved image), our learning scheme can be regarded as weakly-supervised learning where only the class of the observed object is...
weakly labeled but its pixelwise region is not annotated. However, since the retrieval process is not perfect in reality, our proposed method is required to select images where an object of the retrieval keyword is observed, as described in Sect. 3.3.

3.3 Select Correct Images After Instance Segmentation

For semi-supervised learning at \(i\)-th training iteration (\(i > 0\)), Mask-RCNN trained by \((i-1)\)-th iteration is used to detect and segment instances observed in the unsupervised images. As Fig. 5 shows, if the estimated class of a detected instance is not the same as the keyword, this instance region is not used for creating additional training images (ATIs).

Even if the estimated class of a detected instance is the same as the keyword, the estimated class is possibly incorrect; for example, an onion observed in an image retrieved by the keyword “kiwi” can be regarded as kiwi by the failure of Mask-RCNN. For selecting object regions each of whose class is correctly detected, our proposed method evaluates the reliability of instance segmentation. Since Mask-RCNN provides the class confidence of each detected object, this confidence value is employed as the reliability of object detection.

While the aforementioned class confidence is employed also in traditional self-training methods for object detection and recognition, Mask-RCNN sometimes gives a high confidence value unsuccessfully to unexpected regions. Figure 7 shows some examples of abnormal silhouettes that must be rejected. Two images in the left side show small parts of target objects. Two images in the right side show the overlaps of multiple instances (i.e., cucumbers and bananas in the left and right images, respectively). These small parts and overlaps must be rejected for updating the model of Mask-RCNN as described in Fig. 1.

For more reliable correctly-detected region selection, the silhouette of each detected region is used as an additional cue in our method. In order to reject anomalously-silhouetted regions such as those in Fig. 7, we use anomaly detection with Auto Encoder (AE) [32]. In the training process for this procedure, the AE of each object class is trained with silhouette images observed from various orientations in the 3D CG simulator. As the loss function for this training, the MSE loss is used. For rejecting anomalously-silhouetted regions, each extracted region is fed into the AE in order to calculate the reconstruction error. If the reconstruction error is larger than a threshold, the region fed into the AE is rejected.

The threshold is determined by the mean, \(\mu\), and the variance, \(\sigma\), of the reconstruction error values of all training (correct) images in each class. Assuming that the distribution of this reconstruction error follows a Gaussian distribution, the threshold is \(\mu + 3\sigma\) so that about 99% of the training images are classified to be normal.

Note that Mask-RCNN also evaluates the silhouette of an instance as well as its textures. While Mask-RCNN tries to generalize the silhouettes of training data, this leads to increased false-positives. Instead, our strategy is to strictly limit the acceptable silhouettes of a target object based on the shape of its 3D CG model. It can be also mentioned that our method fully employs available data (i.e., (1) instance segmentation model trained by all RGB images and (2) 3D CG models) for this selection process.

3.4 Create Additional Training Images

All regions that are not rejected by the AE are regarded as images whose class and silhouette are correctly detected. These regions are called “additional instances” (AIs) for updating Mask-RCNN.

While the initial model is trained by images synthe-
sized from 3D CG models, we have to synthesize ATIs from the AIs. Based on the inspiration from domain randomization [21], our method synthesizes the ATIs as follows:

- **1. Determine the number of ATIs:** The number of the AIs is imbalanced among object classes because all of image retrieval and correctly-detected instance selections are done independently among the classes. Our method balances the AIs among the classes by upsampling and downsampling. The number of AIs sampled from each class is determined to be $\alpha N_{\text{min}}$ where $\alpha$ is a constant value. $N_{\text{min}}$ denotes the number of newly-detected AIs in the class where the minimum instances are detected at this iteration. In our experiments, $\alpha = 30$ for balancing over-fitting avoidance and sample insufficiency.

- **2. Sample a background image and AIs:** $\alpha N_{\text{min}}$ AIs are randomly and evenly sampled in each class; if $\alpha N_{\text{min}} < N_c$, where $N_c$ denotes the number of newly-detected AIs in $c$-th class, each AI is sampled at most only once and (2) if $\alpha N_{\text{min}} > N_c$, each AI is sampled $\lfloor \frac{\alpha N_{\text{min}}}{N_c} \rfloor$ or $(\lfloor \frac{\alpha N_{\text{min}}}{N_c} \rfloor + 1)$ times. Multiple AIs are pasted onto a background image for efficient learning in our method. The background image is also randomly sampled.

- **3. Create images with randomized AIs and background images:** Sampled AIs are rescaled and rotated for data augmentation. Each rescaled and rotated AI is pasted in a random position in its background image.

The aforementioned procedure creates $\alpha N_{\text{min}} N_{\text{class}}$ ATIs, where $N_{\text{class}}$ denotes the number of all object classes.

### 3.5 Update the Model until Convergence

With $\alpha N_{\text{min}} N_{\text{class}}$ ATIs created at $i$-th iteration, Mask-RCNN is updated. However, Mask-RCNN is updated not only with these $\alpha N_{\text{min}} N_{\text{class}}$ ATIs but also with all ATIs created at $(i-1)$ iteration and former except 0-th iteration; all images used for 0-th iteration are those created from 3D CG models. The motivation of this strategy is that (1) Mask-RCNN is fine-tuned from the initial model trained only by 3D CG models to the model trained for real-world images, (2) Mask-RCNN is not overfit to ATIs created only at later iterations, and (3) since only a small number of 3D CG models are used in our assumption, if we use a large number of images generated from these 3D CG models for iterative finetunings, that gives a negative impact on the finetunings. With this training strategy, we avoid over-fitting of Mask-RCNN to ATIs created at later iterations.

Updating the model described above is iterated until convergence. The iterative learning is halted if the mean Average Precision of mask (mask mAP) decreases in validation data. In order to avoid the effect of the order of iterations (i.e., which images are used at each iteration), Mask-RCNN is finetuned from the initial model with all collected ATIs after convergence.

### 4. Experiments

#### 4.1 Datasets and Models

For creating synthesized images by domain randomization [1], [20], the 3D models of target objects, background images, a variety of texture data for distractors, and a 3D CG simulator are needed. 3D object models are created by Recap [30]. We used about 100 images for each model. The number of object classes is 15; see Table 1. Each class has only one 3D model. Figure 8 shows examples of the

---

**Table 1** The number of retrieved images by keyword search for each of 15 object classes.

|        | apple | avocado | banana | broccoli | carrot | cucumber | garlic | kiwi | mushroom | onion | orange | potato | sweet corn | sweet potato | taro |
|--------|-------|---------|--------|----------|--------|----------|--------|------|-----------|-------|--------|--------|------------|-------------|------|
|        | 1289  | 1564    | 1591   | 1377     | 1497   | 1489     | 1593   | 1517 | 1651      | 1592  | 1211   | 1318   | 1594       | 1546        | 1453 |

---

**Fig. 8** Examples of 3D CG models. In each row, images synthesized from multi viewpoints are shown. From the top to the bottom, carrot, cucumber, onion, orange, and taro are shown.
created 3D CG models. In our experiments, only one 3D CG model was used for each object class. For background images and distractor's textures, Place365 [33] and Dtd [34] datasets were used, respectively. We used Blender as a 3D CG simulator.

With the aforementioned data, the initial model of Mask-RCNN [14], [35] was trained with 10k images synthesized by Blender. Images for semi-supervised learning were retrieved by Google keyword search [36] with Japanese and English keywords. The number of all retrieved images is shown in Table 1. The AE for correctly-detected instance selection was the CAE proposed in [37]. Background images for ATIs were provided by the Place365 dataset [33].

200 test images were captured in our lab setting and manually annotated for instance segmentation. All of these images are not contained in images used for any training processes. The number of foods included in the test dataset is 2 at least and 6 at most for each class. Several examples are shown in Fig. 9. Our 3D object models and test images are available at https://github.com/Obat2343/

† One may use a large number of supervised-training images given by image datasets such as MS COCO. However, these datasets have only some foodstuff classes. Therefore, we did not use these image datasets for 0-th iteration in our proposed method.

4.2 Results of Instance Segmentation

Our method is compared with the following two alternative methods. The first one is Mask-RCNN trained without additional training data, which is called W-ST (Without Self-Training). W-ST was trained by only images synthesized by 3D CG models. These images were created in step 1. Therefore, this is the same as the method called “domain randomization” [1], [20]. The other one is Mask-RCNN trained with ATIs selected based only on class confidence provided by Mask-RCNN (“Class Confidence Selection”: left side of step 3 in Fig. 2). This method is called C-ST (Class only Self-Training). Since C-ST uses only a class confidence, this method skips “Silhouette Anomaly Detection” (right side of step 3 in Fig. 2) of our proposed method. C-ST is the same as traditional “Self-Training” [22].

Table 2 shows experimental results. The results are evaluated with three metrics, namely the mean Average Precision of mask (mask mAP), recall, and precision. These metrics are used with some different parameters. To calculate mAP, the threshold of Intersection-over-Union (IoU) is changed. Recall and precision are used with different
Table 2  Comparative experiments. Mask-RCNN of W-ST-\( p \) was trained with \( p \times 10 \) k images which were created in step 1. In C-ST-\( q \) and Ours-\( q \), Mask-RCNN was trained by \( q \) training iterations of semi-supervised learning. W-ST-1 is equal to the initial model for C-ST and Ours. IRN-coco and IRN-cg indicate that the models are finetuned from coco-pretrained model and W-ST-1 respectively. The best results in each metric are bolded.

|          | W-ST-1 | W-ST-2 | W-ST-3 | C-ST-1 | C-ST-2 | C-ST-3 | C-ST-4 | Ours-1 | Ours-2 | Ours-3 | Ours-4 | IRN-coco | IRN-cg |
|----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|----------|--------|
| mAP50    | 78.9   | 79.9   | 79.6   | 88.2   | 85.9   | 88.6   | 86.4   | 89.3   | 88.9   | 88.2   | 88.3   | 52.8     | 72.3   |
| mAP70    | 76.9   | 77.9   | 77.9   | 85.7   | 82.6   | 86.1   | 83.3   | 87.0   | 86.1   | 85.2   | 85.2   | 47.6     | 66.6   |
| mAP90    | 38.3   | 39.1   | 38.8   | 41.5   | 40.0   | 42.4   | 39.8   | 43.3   | 43.4   | 43.4   | 43.4   | 6.4      | 17.8   |
| recall0.5 | 47.9   | 48.8   | 47.8   | 51.1   | 49.4   | 50.8   | 48.1   | 53.0   | 52.2   | 53.1   | 53.0   | 7.6      | 25.8   |
| recall0.8 | 74.5   | 75.4   | 76.1   | 84.6   | 82.2   | 84.6   | 82.2   | 86.6   | 86.6   | 85.4   | 85.4   | 27.2     | 58.6   |
| precision0.5 | 65.1 | 67.9   | 66.6   | 74.5   | 77.1   | 76.8   | 77.0   | 75.6   | 76.9   | 75.6   | 76.3   | 74.2     | 72.3   |
| precision0.8 | 38.3   | 40.6   | 39.1   | 41.8   | 43.5   | 43.3   | 42.1   | 44.0   | 44.6   | 44.6   | 45.0   | 12.4     | 25.3   |
| precision0.8 | 76.9 | 81.8   | 79.0   | 87.2   | 86.3   | 86.1   | 87.4   | 85.4   | 86.2   | 87.7   | 87.5   | 90.0     | 87.6   |

Confidence thresholds. In Table 2, a subscript and superscript attached to mAP, recall, and precision denote the thresholds of IoU and the confidence value, respectively. For example, mAP_{0.5} denotes that the IoU threshold is 0.5, and recall\(_{0.8}^{50}\) denotes that the IoU and confidence thresholds are 0.5 and 0.8, respectively. Several examples of detections are shown in Fig. 9.

C-ST outperforms W-ST overall. This result validates the effectiveness of self-training. Furthermore, compared to W-ST and C-ST, our method has the best performance in most metrics. Let us consider why our proposed method outperforms C-ST. If the silhouettes and textures of an instance downloaded from the Internet are significantly different from the 3D CG models of any target objects, this instance is rejected in step 3 because its reliability is low. While this process rejects some instances of the target objects, those of non-target objects can also be rejected. If these instances of the non-target objects are trained, the performance of Mask-RCNN is decreased. Our strategy is to train only highly-probable instances of the target objects in order to suppress false-positive detections. Table 2 shows that our image selection using two criteria (i.e., class confidence given by Mask-RCNN and anomalous silhouette detection by AE), Ours-\( q \) in Table 2, is better than the one based only on the class confidence, C-ST-\( q \), except only one metric (i.e., precision\(_{0.8}^{50}\)).

Unfortunately, the performance of our method decreases after two iterations. This is due to the failure of rejecting anomalous instances like Fig. 12. Figures 10, 11, and 12 show examples of correctly-accepted, correctly-rejected and falsely-accepted detected instances, respectively. Figure 12(a) shows typical examples where the silhouette of each instance is similar to that of any object class (denoted by class \( A \)) but the real class of this instance is not class
A. Figure 12 (b) shows another typical example of falsely-accepted instances. In (b), since multiple instances of the same object class (denoted by class B) overlap, their texture patterns are equal to those of class B. In addition, the silhouette of the overlap is similar to that of class B. This kind of falsely-accepted instances are difficult to be rejected by our proposed method. To resolve this problem, we should employ further cues as well as the class confidence of Mask-RCNN and the silhouette of the instance for correctly-detected instance selection.

We compare our method with weakly-supervised instance segmentation, IRN [27], which only needs the class label of each image for training. Since we collect images by keyword search, we use the keyword as a label of each image. Table 2 shows the result of IRN. In both of IRN [27] and our proposed method, it is difficult to learn the correct images of targets because web images include many images where the keyword object is not observed. Furthermore, several target instances are piled up in the image. It makes hard to learn the correct silhouette of each instance. However, our method can reject false-positive instances based on a class confidence given by Mask-RCNN and anomalous silhouette detection by AE.

4.3 Effect of Final Finetuning

As described in Sect. 3.5, our proposed model is finetuned from the model which is trained at 0-th iteration (W-FT-1) after convergence. For verifying the effect of this final finetuning, models with and without it are compared. The model without it is acquired as a result of iterative finetunings; if the iteration is halted at $M$-th iteration, $(M − 1)$-th model is finetuned with images collected at $M$-th iteration, and this finetuned model is used for evaluation. We call the model trained without this Final FineTuning "(C-ST or Ours)-NoFFT."

Table 3 shows experimental results. The result shows that the final finetuning increases the performance in both C-ST and Ours. This might be due to avoiding a bad model drift. In the model finetuned only iteratively (i.e., in (C-ST or Ours)-NoFFT), if some instances are wrongly annotated and used for finetuning, more instances will be wrongly annotated in next iterations. Therefore, the model is progressively overfit to wrongly-annotated images. While our model is also finally finetuned with all images including correctly- and wrongly-annotated images, progressive overfitting to wrongly-annotated images detected at later iterations can be avoided.

4.4 Collected Additional Instances (AIs)

The number of total AIs at each iteration is shown in Table 4. It can be seen that the AIs can be increased by iterative processes. The biggest increase is observed at the first iteration. This effect appears a big performance increase from the initial model (i.e., W-ST-1) to the first iteration (i.e., Ours-1) as shown in Table 2.

5. Concluding Remarks

Our proposed method combines image synthesis and semi-supervised learning to achieve instance segmentation from
a small amount of 3D models and a large number of images on the Internet without manual annotations.

Our future goal with the proposed instance segmentation is a food manipulation task by a robot. For such manipulation, both class prediction and silhouette prediction must be high enough. From a viewpoint of quantitative performance, our estimation of the required performance is 80% for both recall and precision at IoU 0.9. Unfortunately, this goal is not achieved yet. For this goal, our future work is described in what follows.

Since the silhouette of each instance is evaluated by comparing it only with a limited amount of the 3D object models, which are used for initial model training, we have difficulty in applying this method to objects having a variety of 3D shapes. In particular, foodstuffs are reshaped during cooking (e.g., cutting). We plan to cope with this problem by temporally tracking the change of object silhouettes in unsupervised videos. By starting tracking from an instance that can be detected by our proposed method, we can collect the silhouette variety of this object. All of these collected images can be used for finetuning the model of instance segmentation.

References

[1] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, “Training deep networks with synthetic data: Bridging the reality gap by domain randomization,” CVPR Workshops, pp.969–977, 2018.

[2] C. Rosenberg, M. Hebert, and H. Schneiderman, “Semi-supervised self-training of object detection models,” WACV/MOTION, pp.29–36, 2005.

[3] S. Vijayanarasimhan and K. Grauman, “Large-scale live active learning: Training object detectors with crawled data and crowds,” IJCV, vol.108, no.1–2, pp.97–114, 2014.

[4] X. Chen, A. Shrivastava, and A. Gupta, “NeX: Extracting visual knowledge from web data.” ICCV, pp.1409–1416, 2013.

[5] X. Chen and A. Gupta, “Weakly supervised learning of convolutional networks,” ICCV, pp.1431–1439, 2015.

[6] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” CVPR, pp.3431–3440, 2015.

[7] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” MICCAI, pp.234–241, 2015.

[8] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” NIPS, pp.91–99, 2015.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A.C. Berg, “Ssd: Single shot multibox detector,” ECCV, pp.21–37, 2016.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” CVPR, pp.779–788, 2016.

[11] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” CVPR, pp.6517–6525.

[12] J. Dai, K. He, and J. Sun, “Instance-aware semantic segmentation via multi-task network cascades,” CVPR, pp.3150–3158, 2016.

[13] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, “Fully convolutional instance-aware semantic segmentation,” CVPR, pp.2359–2367, 2017.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” ICCV, pp.2961–2969, 2017.

[15] A.Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother, “Instancecut: from edges to instances with multicut,” CVPR, pp.5008–5017, 2017.

[16] D. Novotny, S. Albane, D. Larlus, and A. Vedaldi, “Semi-convolutional operators for instance segmentation,” ECCV, pp.86–102, 2018.

[17] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, “Unsupervised pixel-level domain adaptation with generative adversarial networks,” CVPR, pp.3722–3731, 2017.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. O’zair, A. Courville, and Y. Bengio, “Generative adversarial nets,” NIPS, pp.2672–2680, 2014.

[19] Z. Ren and Y. Jae Lee, “Cross-domain self-supervised multi-task feature learning using synthetic imagery,” CVPR, pp.762–771, 2018.

[20] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” IROS, pp.23–30, 2017.

[21] D. Dwibedi, I. Misra, and M. Hebert, “Cut, paste and learn: Surprisingly easy synthesis for instance detection,” ICCV, pp.1301–1310, 2017.

[22] X. Zhu, “Semi-supervised learning literature survey,” Computer Science, University of Wisconsin-Madison, 2006.

[23] W. Hung, Y. Tsai, Y. Liou, Y. Lin, and M. Yang, “Adversarial learning for semi-supervised semantic segmentation,” BMVC, p.65, 2018.

[24] Y.T. Hu, J.B. Huang, and A. Schwing, “Maskrnn: Instance level video object segmentation,” NIPS, pp.325–334, 2017.

[25] X. Zhao, S. Liang, and Y. Wei, “Pseudo mask augmented object detection,” CVPR, pp.4061–4070, 2018.

[26] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” IEEE Trans. Pattern Anal. Mach. Intell., vol.26, no.9, pp.1124–1137, 2004.

[27] J. Ahn, S. Cho, and S. Kwak, “Weakly supervised learning of instance segmentation with inter-pixel relations,” CVPR, June 2019.

[28] Y. Zhou, Y. Zhu, Q. Ye, Q. Qiu, and J. Jiao, “Weakly supervised instance segmentation using class peak response,” CVPR, pp.3791–3800, 2018.

[29] X. Zhang, Y. Wei, G. Kang, Y. Yang, and T. Huang, “Self-produced guidance for weakly-supervised object localization,” ECCV, pp.597–613, 2018.

[30] AUTOODESK, “Recap.” https://www.autodesk.co.jp/products/recap/

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick, “Microsoft coco: Common objects in context,” ECCV, pp.740–755, 2014.

[32] C. Zhou and R.C. Paffenroth, “Anomaly detection with robust deep autoencoders,” SIGKDD, pp.665–674, 2017.

[33] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” PAMI, vol.40, no.6, pp.1452–1464, 2017.

[34] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, “Describing textures in the wild,” CVPR, pp.3606–3613, 2014.

[35] W. Abdulla, “Mask r-cnn for object detection and instance segmentation on keras and tensorflow,” 2017. https://github.com/matterport/Mask_RCNN.

[36] “Google images download.” https://github.com/hardikvasa/google-images-download.

[37] X. Guo, X. Liu, E. Zhu, and J. Yin, “Deep clustering with convolutional autoencoders,” NIPS, pp.373–382, 2017.
Takeru Oba is a master course student of Toyota Technological Institute, Japan. His research interests are object manipulation by robotic hands and its related technologies such as object detection.

Norimichi Ukita is a professor at the graduate school of engineering, Toyota Technological Institute, Japan (TTI-J). He received the Ph.D degree in Informatics from Kyoto University, Japan, in 2001. After working for five years as an assistant professor at NAIST, he became an associate professor in 2007 and moved to TTI-J in 2016. He was a research scientist of PRESTO, JST during 2002 - 2006. He was a visiting research scientist at Carnegie Mellon University during 2007-2009. His main research interests are multi-object tracking and human pose and activity estimation.