MT-DSSD: Deconvolutional Single Shot Detector Using Multi Task Learning for Object Detection, Segmentation, and Grasping Detection

Ryosuke Araki[1], Takeshi Onishi[1], Tsubasa Hirakawa[1], Takayoshi Yamashita[1], and Hironobu Fujiyoshi[1]

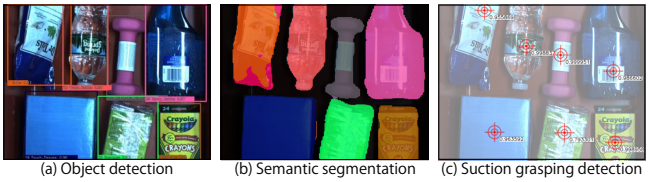
***Abstract*— This paper presents the multi-task Deconvolutional Single Shot Detector (MT-DSSD), which runs three tasks—object detection, semantic object segmentation, and grasping detection for a suction cup—in a single network based on the DSSD. Simultaneous execution of object detection and segmentation by multi-task learning improves the accuracy of these two tasks. Additionally, the model detects grasping points and performs the three recognition tasks necessary for robot manipulation. The proposed model can perform fast inference, which reduces the time required for grasping operation. Evaluations using the Amazon Robotics Challenge (ARC) dataset showed that our model has better object detection and segmentation performance than comparable methods, and robotic experiments for grasping show that our model can detect the appropriate grasping point.**

I. INTRODUCTION

Can robots perform tasks in place of human operators? Also, what is the "visual function" necessary for such work? Grasping detection is a key function for robots that perform manipulative tasks, such as industrial and human assistive robots. Such robots need to recognize the class of the target object and/or grasp it in accordance with what the user or operator orders. From the viewpoints of recognition cost and system running time, object detection and grasping detection should be performed simultaneously and processed at low computational cost and high speed.

Due to the rapid progress of deep learning, methods of object detection and grasping detection achieve better performances than conventional machine learning methods. In the Amazon Picking Challenge 2016 and the Amazon Robotics Challenge (ARC), many teams [1], [2] used deep learning for object detection and grasping detection, and several teams used semantic segmentation for recognizing the detailed area of an object [3], [4], [5]. In the ARC, it is important to detect a specified object from many classes, perform segmentation to recognize the object region in detail, grasp the target object, and put it in delivery cardboard. Furthermore, this task should be performed as quickly and as accurately as a human operator. This applies not only to Amazon's logistics warehouse (the setting assumed in the ARC) but also to ordinary living environments. When the user of the robot wants an object, the robot first detects and recognizes the items in the room and then determines which object the user wants. If the robot finds it, it grasps it securely and delivers it to the user.

[1]They are with Chubu Univ., Aichi, JP {ryorsk@mprg.cs, onishi@mprg.cs, hirakawa@mprg.cs, takayoshi@isc, fujiyoshi@isc}.chubu.ac.jp



(a) Object detection (b) Semantic segmentation (c) Suction grasping detection

Fig. 1. Detection and classification results of proposed model. Our model receives a scene image and then runs (a) object detection, (b) semantic segmentation, and (c) grasping detection for a suction cup.

In this paper, we propose a network model based on a convolutional neural network (ConvNet) that simultaneously performs object detection, semantic segmentation, and grasping detection for a suction cup (Fig. 1). Simultaneous execution of object detection and segmentation by multi-task learning improves the accuracy of both tasks. Additionally, the model detects grasping points and performs the recognition task necessary for robot manipulation. As our model is based on fast object detection, it can perform fast inference. It can also reduce the time required for grasping operation.

II. RELATED WORK

Object detection and semantic segmentation are fundamental tasks in the computer vision field. These tasks are also often used in the robotics field. Object detection is used to detect the objects to be grasped by a robot. Semantic segmentation is used for understanding the surrounding environment and for detecting a more detailed object area than simple object detection. By combining these methods with a grasping detection method, the robot can grasp a specific object.

A. Object Detection with SSD and DSSD

The Single Shot Multibox Detector (SSD) [6] detects object fast by a single network. With its six prediction modules, it detects object candidates of various shapes using a rectangle called a default box that has multiple aspect ratios. Offset for each default box is obtained by regression from feature maps (localization maps) in order to surround the objects with the rectangle. Similarly, the class of the object is recognized using feature maps (confidence maps) obtained by the prediction module representing the likelihood. For all object candidates obtained by this processing, excessive object candidates of the same class are deleted by non-maximum suppression (NMS) processing to reduce excessive detection.

The Deconvolutional Single Shot Detector (DSSD) [7] is an extension of SSD to which deconvolution layers have been

added. By up-sampling the feature map using deconvolution processing, DSSD can obtain richer features through the deconvolution layers than the one by SSD. These deconvolution layers combine the feature maps obtained from the convolution process and the feature maps obtained from the deconvolution process using the deconvolution module. Prediction using the combined feature maps makes the object detection multi-scale, which has the advantage of making SSD more accurate.

B. Semantic Segmentation

The typical approach for semantic segmentation has the structure of an encoder-decoder network. For example, SegNet [8] performs the segmentation with an encoder-decoder structure, where the encoder repeats convolution and max pooling and the decoder repeats upsampling and convolution and expands the feature maps to the input size.

C. Multi-task learning

Executing multiple tasks simultaneously with a single ConvNet can reduce computational costs and improve the accuracy of each task [9]. In particular, executing object detection and semantic segmentation simultaneously [10], [11] improves the accuracy of each task, and also recognizes each object when multiple objects of the same class exist. In other words, this model delivers results close to instance segmentation. This mechanism is necessary for grasping detection in scenes where multiple objects of the same class may exist.

D. Grasping Detection using Deep Learning

Following the appearance of the well-known ConvNet "AlexNet[12]", many methods using deep learning have been proposed. The grasping detection methods use supervised deep learning [13], [14], [15], [16] to learn the grasping points and the features of the object by giving the correct grasping points or area to the learning images as supervisor data in advance. These methods can detect various types of objects, such as non-rigid objects whose appearance changes, which are difficult to detect with CAD model fitting, or transparent objects with point clouds that cannot be acquired.

Lenz *et al.* proposed grasping detection using a two-stage deep neural network (DNN) [13], combining DNN with the two-stage grasping detection [17] proposed by Jiang *et al.* This method narrows down the number of grasping candidates by two processing steps using two DNNs of different scales. Redmon & Angelova [14] proposed a method for detecting grasping points using ConvNet. Their method splits the input image into 7×7 grids and outputs the grasping points and the confidence for each grid by AlexNet-based ConvNet.

In order to reduce data collection and manual labeling work, methods of acquiring learning data [18], [19] and grasping operation [20] by robots have been proposed. These methods use real robots and are highly reliable. They not only detect grasping points better than when using human-labeled data but also estimate the robot's motion, which enables real robots to perform tasks directly. However, these methods take too much time and require many robots, and the learning model cannot be used when the configuration of the robot changes drastically.

Mahler & Goldberg created a 3D object model database called "Dex-Net" [21], [22], [23], [24] in which grasping points are labeled as 3D models. They also proposed a ConvNet model for this data. Labels for grasping points are calculated from the robot hand model and environmental information. It does not require manual annotation.

Many studies have pointed out the usefulness of the suction gripper. In the Amazon Picking/Robotics Challenge, many teams had success using the suction gripper to pick up various everyday items. The suction gripper outperforms the multi-finger gripper in situations where the area of approach is narrow, and objects must be picked up at a single contact point. For stable grasping, a robot can approach the plane area and make a vacuum between the object and the gripper.

However, for some objects, grasping detection is difficult using only the depth image. For example, objects with colorless glass, plastic, and mesh surfaces pass light, so depth values are missing in a typical depth sensor. In addition, a sensor using visible light cannot obtain the depth value of objects that have a gloss or absorb light. Researchers have addressed this with a method for grasping detection from the semantic segmentation result [5], a method to output semantic segmentation and graspability at the same time [25], and a method for grasping detection using an RGB-D image at the same time [14].

III. Multi-task DSSD

As the structure of DSSD is similar to the encoder-decoder structure of SegNet, our concept is to have multi-task DSSD (MT-DSSD) improve accuracy by learning object detection and semantic segmentation at the same time. Furthermore, we add a grasping point detector for the suction cup to this model. The network structure of our MT-DSSD is shown in Fig. 2. We use the VGG[26]-based DSSD as a base and add a layer that performs semantic segmentation and object grasping detection. As a result, the three tasks (object detection, semantic segmentation, and grasping detection) are performed simultaneously using the common feature maps.

A. Deconvolution Module and Prediction Layer

In the deconvolution module, as shown on the left side of Fig. 3, feature maps from the previous layer and the encoder are merged by an element-wise sum. After that, feature maps are input to the activation function (ReLU). At this time, the feature maps of the previous layer are expanded to the same size as the feature map of the encoder by deconvolution. After processing by the deconvolution module, object detection, semantic segmentation, and grasping detection are performed at the same time by three branches in the prediction module, as shown on the right side of Fig. 3.

1) Detection branch: As in the original SSD and DSSD, the detection branch outputs the localization maps, which are offsets to the default box, and the confidence maps, which are the class likelihood of each default box.

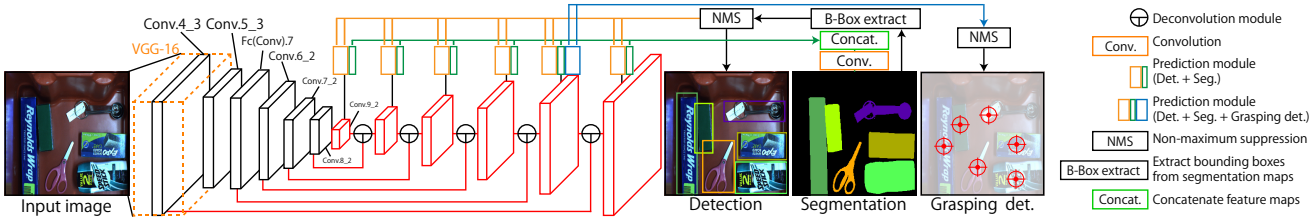


Fig. 2. Model details of the proposed MT-DSSD. We add a layer for semantic segmentation and grasping detection to the DSSD. Thanks to multi-task learning, the object detection and semantic segmentation are accurate, and grasping points are detected at the same time.

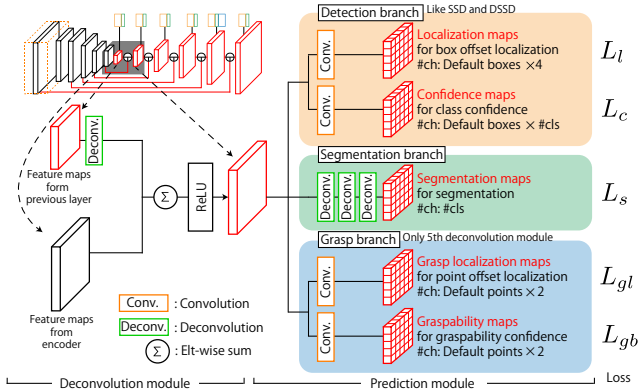


Fig. 3. Deconvolution module and prediction module.

2) Segmentation branch: The segmentation branch outputs the segmentation maps with three deconvolution layers and expands them to the input size. After that, all segmentation maps obtained by the six prediction modules are concatenated and convoluted, and we obtain the final semantic segmentation result.

3) Grasp branch: The grasp branch regresses the grasping point (x_{ij}, y_{ij}) for each pixel of the feature map in order to detect the correct grasping point for the suction cup. We call this the "default point". The number of default points of the six prediction modules is 1,940, which is equal to the number of pixels of all feature maps from each branch. These default points are too numerous for grasping detection of the whole image, so we use only the fifth prediction module (19×19, 361 default points) from the input side. Detecting the grasping point with the minimum necessary feature maps helps keep the whole network from slowing down. We call the feature maps that regress the default points "grasp localization maps". Similar to localization maps in the detection branch, grasp localization maps obtain the offset of default points and bring the default points close to the correct grasping points. We call the feature maps that represent the graspability for each default point "graspability maps". The model learns the positive grasping points as 1 and the rest as 0 for computing graspability.

B. Loss Function

Our model optimizes the following loss function:

$$L(\cdot) = \alpha_c L_c(\mathbf{x_b}, \mathbf{c_b}) + \alpha_l L_l(\mathbf{l_b}, \mathbf{g_b}) + \alpha_s L_s(\mathbf{c_s}, \mathbf{g_s}) + \alpha_{gl} L_{gl}(\mathbf{l_g}, \mathbf{g_g}) + \alpha_{gb} L_{gb}(\mathbf{x_g}, \mathbf{c_g}). \quad (1)$$

$L_c(\cdot)$ is the class probability loss for the confidence maps, which is the same as the original SSD. The loss is the softmax loss over multiple class confidences $\mathbf{c_b}$ and is computed by

$$L_c(\mathbf{x_b}, \mathbf{c_b}) = -\frac{1}{N} \begin{cases} \sum_{i \in (x_{b_i} \neq 0)}^{N} x_{b_i}^p \log(\hat{c}_{b_i}^p) & \text{(each 4 epoch)} \\ \sum_{i}^{N} x_{b_i}^p \log(\hat{c}_{b_i}^p) & \text{(otherwise)} \end{cases}, \quad (2)$$

where $\hat{c}_{b_i}^p$ is the output of a softmax function and $\mathbf{x_b} \in \{0, 1, \ldots, \#\text{class}\}$ is the correct classification class. $\hat{c}_{b_i}^p$ represents the i-th probability of the class p. As with SSD[6], most of the learning data are negative samples. Therefore, hard negative mining is performed to reduce the imbalance between positive and negative samples. Furthermore, we found empirically that model optimization is faster by not learning the background class every four epochs.

$L_l(\cdot)$ is the object candidate loss for the localization maps. Since the object candidate area estimation corresponds to regressing the estimated coordinates to the correct coordinates, this is shown using mean square error, as

$$L_l(\mathbf{l_b}, \mathbf{g_b}) = \frac{1}{4N} \sum_{i}^{N} \sum_{m \in cx, cy, w, h} (l_{b_i}^m - \hat{g}_{b_j}^m)^2, \quad (3)$$

where $\mathbf{l_b}$ is the estimated offset of default boxes and $\mathbf{g_b}$ is the correct offset of default boxes. cx and cy indicate the center location of the box. w and h indicate the width and height of the box.

$L_s(\cdot)$ is the semantic segmentation loss for the segmentation maps. Since the segmentation is a classification task, the loss is the softmax loss over multiple segmentation confidences $\mathbf{c_s}$ and is computed by

$$L_s(\mathbf{c_s}, \mathbf{g_s}) = -\frac{1}{N} \sum_{i}^{N} g_{s_i}^p \log(\hat{c}_{s_i}^p),$$

where $\hat{c}_{s_i}^p$ is the output of a softmax function and $\mathbf{g_s}$ is the correct class of semantic segmentation.

$L_{gl}(\cdot)$ is the grasping candidate loss for the grasp localization maps. Like localization maps, this is shown using mean square error, as

$$L_{gl}(\mathbf{l_g}, \mathbf{g_g}) = \frac{1}{2N} \sum_{i}^{N} \sum_{m \in cx, cy} (l_{g_i}^m - \hat{g}_{g_j}^m)^2, \quad (4)$$

where $\mathbf{l_g}$ is the estimated offset of default points and $\mathbf{g_g}$ is the correct offset of default points. cx and cy indicate the coordinates of default points.

$L_{gb}(\cdot)$ is the graspability loss for the graspability maps. This classifies two classes: "grasping possible" and "grasping impossible". Therefore, the loss is the softmax loss over multiple graspability confidences $\mathbf{c_g}$ and is computed by

$$L_{gb}(\mathbf{x_g}, \mathbf{c_g}) = -\frac{1}{N}\begin{cases}\sum_{i\in(x_{g_i}\neq 0)}^{N} x_{gi}^{p_g}\log(\hat{c_{gi}}^{p_g}) & \text{(each 4 epoch)}\\ \sum_{i}^{N} x_{gi}^{p_g}\log(\hat{c_{gi}}^{p_g}) & \text{(otherwise)}\end{cases}, \quad (5)$$

where $\hat{c_{gi}}^{p_g}$ is the output of a softmax function. $\mathbf{x_g} = \{0, 1\}$ is the correct class of graspability. $\hat{o}_i^{p_g}$ represents the i-th probability of graspability, $p_g = 1$ is the probability that it can grasp, and $p_g = 0$ is the probability that it cannot grasp.

We set each value of α in Eq. (1) as $\alpha_c = 0.25$, $\alpha_l = 0.25$, $\alpha_s = 0.25$, $\alpha_{gl} = 0.125$, and $\alpha_{gb} = 0.125$. These weights were determined empirically, the same as the hyperparameters of the network. In general, we found that it is better to set a smaller weight for tasks where the loss converges quickly, or where the loss value is larger than other tasks. If these α values are not appropriate, learning will not be stable, and loss will diverge.

C. NMS reflecting segmentation result

To obtain a more accurate object bounding box, the non-maximum suppression (NMS) of the object detection reflects the result of semantic segmentation. First, the bounding rectangle of each object is obtained from the result of semantic segmentation, and we calculate the IoU between the bounding box of the object detection and the bounding rectangle. Next, we calculate the average of the class score of the bounding box, use it as the class score of the new bounding box, and process the NMS.

IV. EXPERIMENTS

A. Dataset

There are many datasets for the object detection task, semantic segmentation task, or both, but there are no datasets for both tasks and the grasping detection task. Also, since we developed this model for the picking task, it is not desirable to evaluate it using a common object detection dataset. For these reasons, we modified and used the Team MC^2 ARC2017 RGB-D Dataset[27] featuring 40 kinds of items used in the ARC2017. It includes 1,210 images for learning and 250 images for evaluation. The evaluation images contain scenes where multiple objects of the same class exist. The dataset and item list can be found at our research group's web site (`http://mprg.jp/research/arc_dataset_2017_e`). For the suction grasping point, we annotated in accordance with rules. As a result, the model can learn grasping points that humans can intuitively grasp, and we don't need to consider geometric properties. In fact, just a few hours were sufficient for manually annotating the grasping positions.

B. Evaluation Metrics

1) Object Detection: We evaluated the object detection by measuring the recognition rate, miss rate, and mean IoU. The recognition rate is the rate at which the detected objects can be correctly recognized. It means that the estimated class is equal to the correct class (# detected objects that have the correct class / # detected objects). The miss rate is the rate at which it could not be detected for all objects (# objects that could not be detected / # objects that should be detected). The mean IoU is the average of the IoU $= (\mathrm{area}(B_d) \cap \mathrm{area}(B_t)) \,/\, (\mathrm{area}(B_d) \cup \mathrm{area}(B_t))$, between the detected bounding box (B_d) and the correct bounding box (B_t).

2) Semantic Segmentation: We evaluated the semantic segmentation by the global accuracy, class accuracy, and mean IoU, the same as in Pascal VOC[28].

3) Grasping Detection: We evaluated the grasping detection by measuring the accuracy, miss rate, and mean distance. Accuracy represents the ratio at which the Euclidean distance on the image (1280 $\times$ 960 pixels) between an estimated grasping point and the correct grasping point is 80 pixels or less. Miss rate represents the ratio at which the grasping points could not be detected for all correct grasping points. Mean distance represents the average of the Euclidean distances of the correct grasping points. Also, in order to determine if the grasping detection result of the proposed model is actually appropriate, we applied our model to a real robot system and performed a grasping experiment.

C. Experimental Results of Object Detection

We compared our object detection models with an original SSD, a single-task model of DSSD that only performs object detection. The evaluation results of object detection are listed in Table I. Our model improved the recognition rate by 11.59 points compared with SSD, thus demonstrating its high accuracy. In addition, our model improved the mean IoU by 2.12 points compared to the single-task DSSD. We conclude from these results that our model can detect the object region with higher accuracy due to using the NMS reflecting the results of semantic segmentation.

Examples of object detection results for each method are shown in Fig. 4(a). In the case of the scene with little overlap of objects (first row in the figure), most objects could be detected. However, depending on the appearance of the object and the overlapping, sometimes the detection was not done well. In the case of a scene with many overlapping objects (second row), sometimes objects hidden by another object were not detected. Overall, our model could detect the object region closer to the correct region than the other models, as with `Bath_Sponge` in the first row and `Knit_Gloves` in the second row.

D. Experimental Results of Semantic Segmentation

Next, we compared our model with three semantic segmentation models: SegNet, U-Net[29], and PSPNet [30]. The evaluation results of semantic segmentation are listed

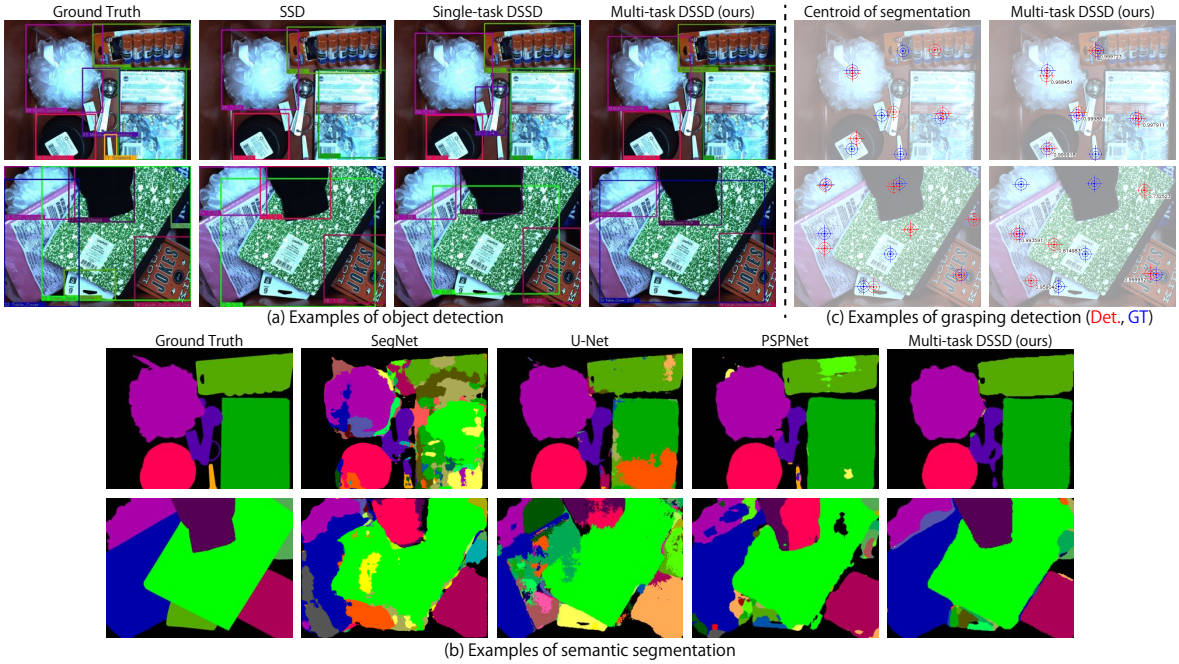


Fig. 4. Examples of results for (a) object detection, (b) semantic segmentation, and (c) grasping detection. (a) When objects were simply arranged, the model performed detection stably. However, it was difficult to detect bulk scenes of multiple kinds of items. (b) The proposed model performed recognition with better accuracy than the other models. Miss recognition of an overlapping area of objects and near the boundary decreased. (c) Comparing the correct labels (blue) with the detection results (red), the proposed model detected grasping points closer to the correct labels. Both methods had difficulty detecting in bulk scenes.

TABLE I

EVALUATION RESULTS OF OBJECT DETECTION (%).

Algorithm	Recog. rate	Miss rate	Mean IoU
SSD	80.67	**24.76**	76.70
Single-task DSSD	87.18	29.01	80.96
Multi-task DSSD	**92.26**	32.91	**83.08**

TABLE II

EVALUATION RESULTS OF SEMANTIC SEGMENTATION (%).

Algorithm	Global Acc.	Class Acc.	Mean IoU
SegNet	72.11	64.30	44.69
U-Net	79.26	69.95	54.19
PSPNet	85.28	79.42	67.76
Multi-task DSSD	**90.49**	**85.04**	**76.01**

in Table II. Our model improved each evaluation index compared to the other models.

Examples of the semantic segmentation results for each model are shown in Fig. 4(b). Compared with the other models, ours could suppress mistakes in classes and avoid miss recognition occurring at the edges of objects and at boundaries with other objects. Our model dramatically reduced the case of recognizing several classes on one object compared to SegNet and U-Net. However, in the case of scenes with many overlapping objects, none of the models could accurately recognize at the boundary with other objects.

E. Experimental Results of Grasping Detection

To the best of our knowledge, there is no method for detecting grasping points for the suction cup from an RGB image. Therefore, for this comparison, we use a case where

TABLE III

EVALUATION RESULTS OF GRASPING DETECTION.

Method	Accuracy (%)	Miss rate (%)	Mean dist. (pix)
Centroid	50.62	37.78	38.27
Multi-task DSSD	**76.18**	**33.08**	**29.11**

the centroid of the semantic segmentation result is set as the grasping points. The evaluation results of grasping detection are listed in Table III. Our model improved the accuracy by 25.56 points and the mean distance by 9.16 points compared with the case using the centroid of the semantic segmentation result. These results demonstrate that our model can detect optimum grasping points close to the correct ones.

Examples of the grasping detection results are shown in Fig. 4(c). When using the centroid of semantic segmentation results, the model detected the grasping point as near the center of the object regardless of the shape or appearance of the object. Therefore, some grasping points were inaccurate, such as `Measuring_Spoons` and `Glue_Sticks` in the second row of the figure. In addition, when the result of semantic segmentation was incorrect, it not only detected incorrect grasping points but also an excessive number of them. We conclude that our model acquired the optimum grasping points for each object by learning. In the first row of the figure, our model avoided the difficult part to grasp on `Measuring_Spoons` and detected the grasping points on the product label made of paper. Also, our model avoided the non-flat surface of `Glue_Sticks` and detected the area of the plane. In scenes with many objects (the second row), our model avoided detecting unnecessary grasping points,

TABLE IV

SUCCESS RATE (%) IN GRASPING EXPERIMENTS (SINGLE ITEM).

Item ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Centroid	N/A	10	50	0	60	**100**	N/A	**100**	50	**100**	N/A	**100**	**100**	**100**	50	80	0	**100**	0	0
Proposed	N/A	**20**	**60**	**80**	**100**	**100**	N/A	**100**	**80**	**100**	N/A	**100**	90	**100**	**90**	**100**	**50**	**100**	**40**	**10**
Item ID	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Centroid	N/A	N/A	0	50	60	**100**	0	**100**	**100**	80	40	90	N/A	10	50	40	N/A	70	N/A	N/A
Proposed	N/A	N/A	**20**	**100**	**70**	**100**	**60**	**100**	**100**	**100**	**100**	**100**	N/A	**80**	**80**	**100**	N/A	**90**	N/A	N/A

TABLE V

INFERENCE SPEED COMPARISON WITH DIFFERENT METHODS.

Task			Processing time [ms]		
Det.	Seg.	Grasp.	CPU	GPU	TX2
√			817	128	589
√	√		1276	256	833
√	√	√	1283	263	851

avoided items with many overlaps, and successfully detected grasping points that were easy to grasp.

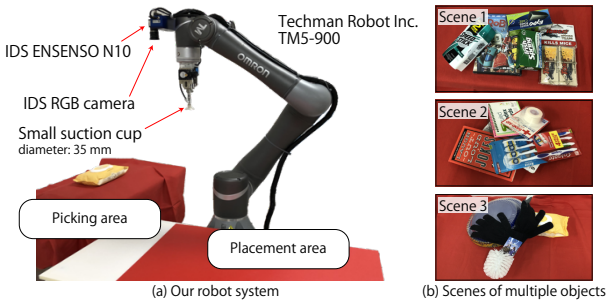
F. Inference Speed Comparison

A comparison of the processing time (excluding screen drawing time) is shown in Table V. The experimental environments were CPU: Intel Core i7 6700K 4.00GHz, GPU: NVIDIA GTX 1080Ti, and TX2: NVIDIA Jetson TX2. Even when our model used TX2, which is an embedded device, it could detect at a sufficient speed for robotic picking. Many methods of general object detection are run at 10 FPS or more, but considering the picking task, even this level is expected to deliver a sufficient performance. Furthermore, our model detects multiple grasping points at once and can make an approach many times with one sensing action as long as the position of the object does not change. For this reason, we consider these detection speed results to be sufficient.

G. Robotic Grasp Evaluation

Simply comparing the results with the correct label is insufficient to evaluate the grasping detection. Even when the detected grasping point is some distance away from the correct label, grasping points might be grasped if they are detected in the object area. Therefore, in this experiment, we evaluated by robotic grasp whether the grasping points detected by our model are appropriate. Our robot system is shown in Fig. 5(a). Our method detects grasping points on the object in the picking area where the red sheet is laid. The robot approaches the normal direction of the nearby point of the obtained grasping points. When the robot picks and places at the destination area located at the side, we deem it a success. However, we exclude some objects from the target of grasping if they exceed the payload of the end effector or do not have an area that can be grasped.

The grasping success rate of each item when putting a single item in the picking area is shown in Table IV. The robot grasps each item ten times, and each item has a different pose every time. We compare our model with a case where the centroid of the segmentation result is used as the grasping position. In both cases, the robot could stably



(a) Our robot system (b) Scenes of multiple objects

Fig. 5. (a) Our robot system and (b) grasping experiments scenes.

TABLE VI

RESULTS OF GRASPING EXPERIMENTS (MULTI-ITEM).

		Scene 1	Scene 2	Scene 3
No. of trials for picking	Centroid	7	9	19
	Proposed	4	5	4
Success rate (%)	Centroid	57	44	11
	Proposed	100	100	75

grasp items with many flat areas. The robot with our model could grasp `Toilet_Brush` (ID: 4) and `Glue_Sticks` (ID: 9) while avoiding difficult suction areas. When using the centroid, it could not grasp some items well.

The number of trials for picking all objects (/ in 10 minutes if picking has not finished) and the grasping success rate of each scene when putting multiple items in the picking area are shown in Table VI. We experimented with the three scenes shown in Fig. 5(b). The robot with our model was able to perform stable grasping even for multiple items.

V. CONCLUSION

In this paper, we have presented multi-task DSSD (MT-DSSD), a model that simultaneously runs object detection, semantic object segmentation, and grasping detection of a suction cup. The proposed model can perform object detection and semantic segmentation with high accuracy while simultaneously detecting the grasping points of the target object. We installed the proposed model in a real robot system and demonstrated through robotic grasp experiments that the detected grasping points are suitable for picking tasks.

Our model, with its ability to detect objects and grasping points at high speed, can be applied to visual servoing that runs path planning in real time while sensing. In the future, we will proceed with research on deep visual servoing.

Acknowledgement. This work was carried out based on the budget of NEDO “next generation artificial intelligence & robot core technology development”.

REFERENCES

[1] C. Hernandez, M. Bharatheesha, W. Ko, H. Gaiser, J. Tan, K. van Deurzen, M. de Vries, B. Van Mil, J. van Egmond, R. Burger, *et al.*, "Team Delft's Robot Winner of the Amazon Picking Challenge 2016," in *Robot World Cup*. Springer, 2016, pp. 613–624.

[2] H. Fujiyoshi, T. Yamashita, Y. Yamauchi, T. Hasegawa, M. Hashimoto, S. Akizuki, Y. Domae, and R. Kawanishi, "Team C^2M: Two Cooperative Robots for Picking and Stowing in Amazon Picking Challenge 2016," in *Warehouse Picking Automation Workshop on International Conference on Robotics and Automation*, 2017.

[3] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, "Multi-view Self-supervised Deep Learning for 6D Pose Estimation in the Amazon Picking Challenge," in *IEEE International Conference on Robotics and Automation*. IEEE, 2017, pp. 1386–1383.

[4] M. Schwarz, A. Milan, C. Lenz, A. Munoz, A. S. Periyasamy, M. Schreiber, S. Schüller, and S. Behnke, "NimbRo Picking: Versatile Part Handling for Warehouse Automation," in *IEEE International Conference on Robotics and Automation*. IEEE, 2017, pp. 3032–3039.

[5] D. Morrison, A. W. Tow, M. McTaggart, R. Smith, N. Kelly-Boxall, S. Wade-McCue, J. Erskine, R. Grinover, A. Gurman, T. Hunn, D. Lee, A. Milan, T. Pham, G. Rallos, A. Razjigaev, T. Rowntree, K. Vijay, Z. Zhuang, C. Lehnert, I. Reid, P. Corke, and J. Leitner, "Cartman: The Low-Cost Cartesian Manipulator that Won the Amazon Robotics Challenge," in *IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 7757–7764.

[6] 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.

[7] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD: Deconvolutional Single Shot Detector," *arXiv preprint arXiv:1701.06659*, 2017.

[8] 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.

[9] S. Li, Z.-Q. Liu, and A. B. Chan, "Heterogeneous Multi-task Learning for Human Pose Estimation with Deep Convolutional Neural Network," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 482–489.

[10] N. Dvornik, K. Shmelkov, J. Mairal, and C. Schmid, "BlitzNet: A Real-Time Deep Network for Scene Understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4154–4162.

[11] J. Cao, Y. Pang, and X. Li, "Triply Supervised Decoder Networks for Joint Detection and Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7392–7401.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012, pp. 1097–1105.

[13] 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.

[14] 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.

[15] Z. Wang, Z. Li, B. Wang, and H. Liu, "Robot grasp detection using multimodal deep convolutional neural networks," *Advances in Mechanical Engineering*, vol. 8, no. 9, p. 1687814016668077, 2016.

[16] D. Guo, F. Sun, T. Kong, and H. Liu, "Deep vision networks for real-time robotic grasp detection," *International Journal of Advanced Robotic Systems*, vol. 14, no. 1, p. 1729881416682706, 2016.

[17] Y. Jiang, S. Moseson, and A. Saxena, "Efficient Grasping from RGBD Images: Learning using a new Rectangle Representation," in *IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3304–3311.

[18] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection," *International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[19] L. Pinto and A. Gupta, "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours," in *IEEE International Conference on Robotics and Automation*. IEEE, 2016, pp. 3406–3413.

[20] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[21] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-Net 1.0: A Cloud-Based Network of 3D Objects for Robust Grasp Planning Using a Multi-Armed Bandit Model with Correlated Rewards," in *IEEE International Conference on Robotics and Automation*. IEEE, 2016, pp. 1957–1964.

[22] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics," *Robotics: Science and Systems*, 2017.

[23] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-Net 3.0: Computing Robust Vacuum Suction Grasp Targets in Point Clouds using a New Analytic Model and Deep Learning," in *IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 1–8.

[24] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning Ambidextrous Robot Grasping Policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.

[25] E. Matsumoto, M. Saito, A. Kume, and J. Tan, "End-to-end Learning of Object Grasp Poses in the Amazon Robotics Challenge," in *Warehouse Picking Automation Workshop, ICRA*, 2017.

[26] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *International Conference on Learning Representations*, 2015.

[27] R. Araki, T. Yamashita, and H. Fujiyoshi, "ARC2017 RGB-D Dataset for Object Detection and Segmentation," in *Late Breaking Results Poster on International Conference on Robotics and Automation*, 2018.

[28] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, jan 2015.

[29] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[30] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2881–2890.