International Journal of Automation and Computing
www.ijac.net

18(2), April 2021, 277-287
DOI: 10.1007/s11633-020-1260-1

Suction-based Grasp Point Estimation in Cluttered Environment for Robotic Manipulator Using Deep Learning-based Affordance Map

Tri Wahyu Utomo Adha Imam Cahyadi Igi Ardiyanto

Department of Electrical Engineering and Information Technology, Faculty of Engineering, University of Gadjah Mada, Yogyakarta 55281, Indonesia

Abstract: Perception and manipulation tasks for robotic manipulators involving highly-cluttered objects have become increasingly in-demand for achieving a more efficient problem solving method in modern industrial environments. But, most of the available methods for performing such cluttered tasks failed in terms of performance, mainly due to inability to adapt to the change of the environment and the handled objects. Here, we propose a new, near real-time approach to suction-based grasp point estimation in a highly cluttered environment by employing an affordance-based approach. Compared to the state-of-the-art, our proposed method offers two distinctive contributions. First, we use a modified deep neural network backbone for the input of the semantic segmentation, to classify pixel elements of the input red, green, blue and depth (RGBD) channel image which is then used to produce an affordance map, a pixel-wise probability map representing the probability of a successful grasping action in those particular pixel regions. Later, we incorporate a high speed semantic segmentation to the system, which makes our solution have a lower computational time. This approach does not need to have any prior knowledge or models of the objects since it removes the step of pose estimation and object recognition entirely compared to most of the current approaches and uses an assumption to grasp first then recognize later, which makes it possible to have an object-agnostic property. The system was designed to be used for household objects, but it can be easily extended to any kind of objects provided that the right dataset is used for training the models. Experimental results show the benefit of our approach which achieves a precision of 88.83%, compared to the 83.4% precision of the current state-of-the-art.

Keywords: Grasping point estimation, household objects, red, green, blue and depth (RGBD) channel image, semantic segmentation, cluttered environment.

Citation: T. W. Utomo, A. I. Cahyadi, I. Ardiyanto. Suction-based grasp point estimation in cluttered environment for robotic manipulator using deep learning-based affordance map. *International Journal of Automation and Computing*, vol.18, no.2, pp.277–287, 2021. http://doi.org/10.1007/s11633-020-1260-1

1 Introduction

The implementation of autonomous warehouse logistics has faced many challenges. Amazon and other marketplace companies, such as Alibaba, have been the main promoter of the usage of artificial intelligence on robots in the logistics industry. In 2017, Amazon started a competition called the Amazon robotics/picking challenge. Such robotics pick-and-place are often called a grasping problem in the robotics research field, and performed by several types of actuators, such as robotic fingers (e.g., [1] and [2]) and dexterous hands[3]. The main challenge lies in the wide variety of products to be packed, with various sizes, shapes, weights, and degrees of fragility. The main task of those problem was to safely plan and execute a grasping action which is sometimes needed to work in cooperation with other robots or people.

Most of the current solutions of the robotic pick-and-place task is done by understanding the object model, which normally follows these three steps: object recognition, pose estimation by matching the model of object, and planning the grasp action. These methods usually need to be fed with handcrafted object features to be able to compute the grasping point proposal. Since such methods rely on prior knowledge to the object model, it tends to fail when dealing with a highly cluttered environment. These failures are caused by the inability of the object recognition task to properly recognize and detect each object, and will be made worse by the novel object's appearance. By increasing the number of objects to be recognized, it will also decrease the performance of the algorithm to match the object with prior known object models. A lot of work is then needed to be performed be-

Research Article
Manuscript received May 14, 2020; accepted September 25, 2020; published online January 8, 2021
Recommended by Associate Editor Hong-Nian Yu
Colored figures are available in the online version at https://link.springer.com/journal/11633

fore the algorithm is even used, i.e., to prepare the object model and represent them in a way that the algorithm could understand. Moreover, it will need a large amount of data to perform well.

In this work, we propose a new approach by integrating the object recognition, pose estimation, and model matching stages and using a deep neural network method which directly infers the pixel-wise probability of picking up the object, called an affordance map, from a red, green, blue and depth (RGBD) input image. This approach is highly inspired by the work of Zeng et al.[4] from the MIT-Princeton team in the Amazon Robotics Challenge 2017 which won the 1st place in the competition.

Compared to the work of Zeng et al.[4], our contributions are two-folds. First, we use a modified deep neural network backbone for the semantic segmentation inputs. It normally produces an image which contains a per-pixel logits value of the predicted classification class, and each class is placed into separate channels. The inferred grasping affordances are subsequently taken from the positive grasping class. Secondly, we incorporate a high speed semantic segmentation to the system, which means our solution has a lower computational time, and the inference can be done in near real-time. The grasp point proposal is finally obtained by fetching the highest affordance value of the prediction output.

2 Related works

There have been several works to solve the problem of grasp point estimation. In this section, we will review several works related to robotic grasping and picking systems, which will be grouped based on the approach taken on their work. For more in depth reviews regarding related works done in this field, refer to [5].

2.1 Model-based grasping

This approach is usually referred to as the traditional approach for autonomous robotic picking and grasping. Generally, this approach follows a three-step solution: object recognition, pose estimation, then followed by model matching to get the proper information to execute the motion planning algorithm. The early work on this approach was done by Zhao et al.[6] which utilized a huge laser scanning camera, and it is limited by a static working environment and a single specific object. Later work by Collet et al.[7] only used a normal color camera and built the object′s 3D model from several images, but it still suffered a huge computational cost needed to infer the 6D grasp pose. The more recent solutions with similar approaches have shown the ability to work in unstructured and cluttered environments[8–12]. These approaches generally suffer from similar problems of high computational cost and are still not robust enough to deal with uncertainty that might occur in the real world implementation.

2.2 Data-driven grasping

Most common approaches in grasp point estimation research employ a learning method in their pipeline. As reviewed in [5], most of them use deep learning to do segmentation or recognition of the objects in the scene which takes training data to be learned, but the later steps are still similar to the model-based grasping approach. Ten Pas et al.[13] extended the works by Herzog et al.[11] to generate some grasping hypotheses on any visible surface. Such similar method are also used in [14, 15]. A detailed review of this category of works can be found in [5, 16].

Recent works on this method[17–19] employed point clouds from RGBD camera to estimate a 6D pose of objects in the scene which can achieve a slightly better rate of success compared to the traditional model-based approach. There was also an attempt to only use a single color image in [20]. These works were then extended in [21, 22] by using synthetic datasets, and yielding better overall performance. These works could achieve a more reliable performance in a more challenging environment, but they still suffered a huge computational cost and complexity and failed to work when new novel objects are appeared.

2.3 Direct grasp proposal inference

Early work on this approach by Saxena et al.[23] tried to infer the grasp point directly from input images using a supervised learning model which was trained using synthetic images to predict 3D grasp position using a multi-view red, green and blue (RGB) channel image camera. Despite being trained using synthetic dataset, the model performed well enough when tested with real-world novel objects, but the variety of the objects was still limited and the environment used in the experiment was highly controlled to conform with the designed algorithm.

Another work by Johns et al.[24] used a deep learning model to predict grasp score probability from several grasp pose proposals which were trained by using synthetic 3D models generated from physics simulations. It was then extended by Lu et al.[25] who used real world objects and datasets. Even though those works performed well in their experiments, they still did not address the implementation in highly cluttered environment which differed significantly in terms of the challenges that need to be solved in the algorithm design. In the work by Zeng et al.[4], this problem of real world implementation was addressed. Their approach was implemented in a highly cluttered environment as it did not need to take into account any object of interest in the scene. Compared to [26], they performed the grasp point generalization on several types of objects using shape affordance, which was based on a histogram of orientation

shape descriptors.

3 Semantic segmentation

In general, a neural network model solving the task of semantic segmentation consists of two parts: 1) backbone model which normally uses a pre-trained image classification model without the fully connected layer which acts as a feature extractor for the input image, then followed by 2) segmentation layer which takes the output of the backbone to generate the pixel-wise dense prediction value. The main drawbacks of using deep learning methods is the need for a large amount of data to even properly do a specific task. This not only takes a large portion of preparation time, but sometimes also requires a really high cost to get the proper datasets. Fortunately, the weight of a deep learning model from a specific task can be reused for a different task with the proper modification and could achieve a good performance even when being trained with lower amounts of data. This process is called transfer learning, and the base model being used is normally an image classification task model. Such a base model which is reused for a different task with transfer learning is usually called the backbone layer.

3.1 Backbone

3.1.1 ResNet

The deep residual learning network, or ResNet[27], is the first successful implementation of very deep neural network models which actually gained higher performance. The ResNet model tried to resolve the problem of the accuracy saturation where the accuracy stays the same even when we make the network deeper, which became the major limiting factor of deep neural network performance. It introduces a module layer that act as the building block of the very deep network and is used repeatedly. The main feature of this building block layer is the usage of a shortcut connection, a connection of layers which skips one or more layers. He et al.[27] use a convolutional layer with 3×3 kernel size as the base weight layer. Based on the original work, ResNet models are expanded to have a higher size and better performance into several model sizes, i.e., ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152. The difference of these model sizes is that the higher size models have more building block layers which generally translate to better overall performance.

3.1.2 EfficientNet

The EfficientNet model[28] tried to tackle a common problem of model scaling in an image classification deep learning model which had not yet been understood well before and also had a high efficiency computation for the model. They proposed to carefully balance a combination of depth, width, and resolution scaling using a method they called compound coefficient. The EfficientNet model was built using a neural architecture search[29, 30] to design a new baseline network, then scale it using the method proposed. This architecture search was done by optimizing the accuracy and number of FLOPS (floating operation per second) which correlates to the efficiency of the model from a baseline model of building blocks called the mobile inverted bottleneck (MBConv)[30, 31], with additional squeeze-and-excitation optimization[32]. This model has achieved state-of-the-art performance at the time of publishing with a significantly lower number of parameters used. From the original works of this model in [28], Tan and Le expanded the original EfficientNet model to have a higher size which translated to better performing models. These expanded EfficientNet models are EfficientNet-b0 through EfficientNet-b8, which was expanded using the compound coefficient strategy.

3.2 Segmentation layer

Segmentation layer acts as a mapping function from the output of the backbone to the output of the entire semantic segmentation model which produces the predicted image data with pixel-wise class mapping, where each prediction class is represented as each channel of the output.

3.2.1 Fully convolutional network (FCN)

The FCN segmentation layer[33], as the name suggests, simply does several layers of convolution operation on the output of backbone and sets the last output layer's number of channel to the corresponding number of classes to be predicted. Long et al.[33] use several backbone models to experiment with: AlexNet, VGG, and GoogLeNet. This segmentation layer is the simplest form of a semantic segmentation model which could easily be implemented with various kinds of backbone models. This model is used in this work since it is the simplest one to implement so that it can be a test model to do debugging on the main training script used for the other models as well.

3.2.2 Pyramid scene parsing network (PSPNet)

The major issue with the previous model is the lack of proper usage of global scene category clues achieved by the backbone model which suffers in terms of performance for the complex scene. This will be especially true in the case of this project as the scene in the environment of grasping is a complex set of objects which is highly cluttered. This problem of proper usage of global scene category clues was then addressed in the work of Zhao et al.[34] with the proposed method (PSPNet). PSPNet extends the traditional dilated FCN[35] with their specially designed global pyramid pooling module. The main idea of the pyramid pooling module is to fuse the features achieved by four different pyramid scales from the backbone. With this strategy, it is possible to separate the feature map of the data into four different sub-regions which form a different representation of the features for

each different location, which could possibly collect different levels of information for each location.

3.2.3 Bilateral segmentation network (BiSeNet)

Yu et al.[36] addressed several problems that might correspond to the performance loss in other models due to scaling, to get a better speed performance. Their preliminary studies conclude that by restricting the input size to reduce computation complexity, it could lead to loss of spatial details which subsequently corrupts the prediction output, especially around the boundaries. They proposed a segmentation layer network called bilateral segmentation network (BiSeNet) which consists of two parts: spatial path (SP) and context path (CP). Spatial path portion of the module is used to preserve the spatial information generated by the backbone which yields a high-resolution feature, and the context path module uses a fast downsampling strategy to obtain a sufficient size of receptive field in the output. The output of both paths is then fed into another module called the fusion feature module (FFM) to combine the output of the two paths efficiently. With this strategy, BiSeNet could achieve state-of-the-art methods for real-time semantic segmentation with the ResNet101 backbone and even surpass the accuracy of the PSPNet model with 105 FPS inference time.

4 Suction grasp estimation

4.1 Dataset

The main dataset used in this work is the original dataset used by Zeng et al.[4] in their project and is publicly available for use[1]. It consists of suction grasping, parallel jaw grasping, and image matching datasets. Nevertheless, we only use the suction grasping dataset.

In the available suction grasping dataset, there are four different image sets for a single scene: color input, depth input, color background, and depth background, as illustrated in Fig. 1, and the extra data of camera pose and camera intrinsics are saved as a text file. The objects used in this dataset are based on the widely available and commonly used household objects, such as scissors, boxed objects, water bottle, books, cloth, plastic wrapped object, etc. The dataset images are captured using an Intel® RealSense™ SR300 RGB-D camera. The depth images are stored as a 16-bit single channel image, which represents a calibrated distance in deci-millimeters (10^{-4} m) and invalid depth is set to 0. The color images are stored as a normal portable network graphic (PNG) image with 3-channel 8-bit data. The dataset also provided the train and test split with a proportion of 4 : 1.

The labels of this dataset are stored as a single channel 8-bit image which represents 3 different class predictions: unsuctionable, suctionable, and unlabeled. The unsuctionable area is the area in the image where it would cause unstable suction and lead to a failed grasp. This area is normally located at the part of object that is too far from the center gravity of the object. The suctionable is the area which needs to be predicted where it is normally near the center of gravity of the object and is more likely to give a successful grasping action. These classes are represented as different values in the label image, that is the unsuctionable area is set to 0, suctionable area is set to 128, and unlabeled area is set to 255, as illustrated in Fig. 1 (e).

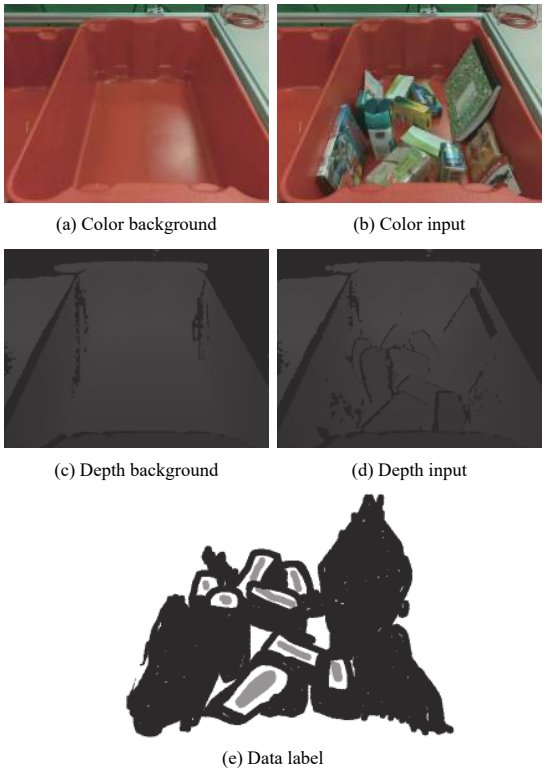


Fig. 1 Dataset sample image of a single scene

4.2 Baseline algorithm

The baseline algorithm for this work is based on the work done by Zeng et al.[4] with several modifications to fit our usage. The general approach of this baseline algorithm is to get the local standard deviation to the surface normals of the pixel to represent the affordance value for the corresponding pixel on an image. It is done by performing a background subtraction to the input image to get an image data that contains objects only in it. Hence, this baseline algorithm also needs the background image of the scene where no objects are in the scene.

The first step to this baseline algorithm is to perform a background subtraction to the input RGBD data, that value has been scaled to have a value range of [0,1]. Subsequently, a threshold value for the background subtraction is set to determine if the particular pixel is a back-

[1] https://vision.princeton.edu/projects/2017/arc/#datasets

ground or not (no change of value on the input compared to background). After performing this operation, we will get a foreground mask for each color and depth image which contains the information of pixel values with only objects in it. We then combine both information in color and depth foreground masks as a single datum using the logical OR operation.

The depth image input is then projected to the camera space using the provided camera intrinsics in the dataset to get the 3D point cloud data of the image input which later could be used to compute the surface normal for each point of the input point cloud. Finally, we are able to compute the local standard deviation from the surface normal by first projecting it back to image plane. The output values of local standard deviation is then normalized to the value range of [0,1], yielding the affordance map value.

4.3 Proposed solution

The general idea of our approach is to generate an affordance map from the input RGBD image of the scene which represents the probability of success of the corresponding pixel, from which we can infer the best grasping proposal from the pixel with the highest probability value (see Fig. 2). We employ a deep neural network method as it is the easiest way to implement because there is no need to manually construct and design any image features specific to the data. It has also performed really well in other tasks. A modified deep neural networks model for semantic segmentation tasks is then used to perform a pixel-wise classification mapping on the output from the input, see Fig. 3 (a). The output of this model is represented as a multi-channel image data in which each channel of the output represents the logits value mapping of class prediction for the corresponding pixel in the input image. It means the number of output channels is the same as the number of classes to be predicted.

There are several strategies to achieve the desired behavior, i.e., by exploiting the semantic segmentation task with the neural networks method. The image classification task model which is used as the backbone model for the semantic segmentation task receives an input of a color image, which means it receives 3-channel of 2D matrix data. To get the best performance, we use an RGBD camera which produces a color image and a depth map or RGBD image. Using a color only image could not convey

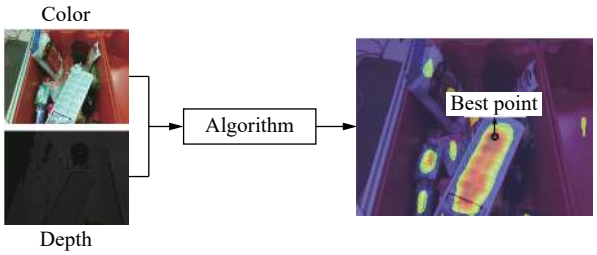
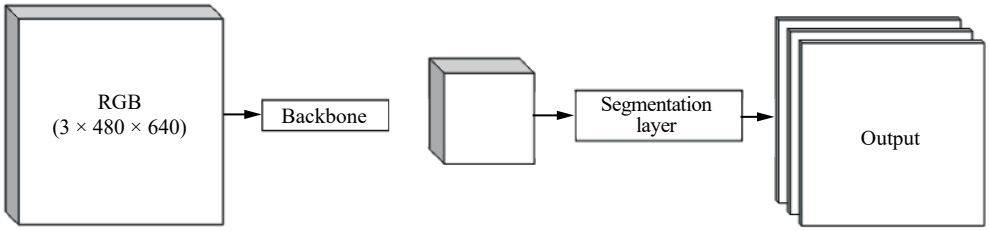
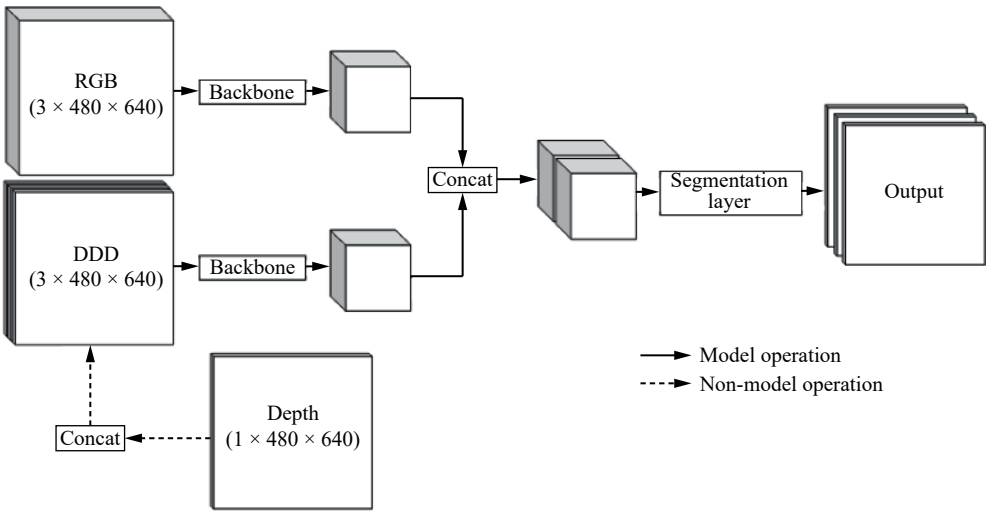


Fig. 2 Illustration of general concept of the proposed solution



(a) Normal semantic segmentation model task illustration



(b) Modified semantic segmentation model proposed

Fig. 3 Comparison of the proposed model with the original model

enough information (features) of the actual surface of the object in the scene. It is especially useful in the case of an object manipulation since the robot will directly interact with the object in the real-world.

To tackle this problem of incompatible input for the model, the backbone of a normal semantic segmentation model, which normally has only a single backbone, is modified to have two different backbone models to compute the features of each input (color and depth) independently of each other. In addition, since a depth image is only a single channel, the depth map data is concatenated to make a 3-channel image. This strategy will then produce two separate output features from backbones, which cannot be processed directly by the segmentation layer. To merge the resulting output features from the backbones, we implement a concatenation operation on the channel dimension which yields an output of twice the size of those normal semantic segmentation layer model. Thus, we also need to increase the input size of the very first operation on every segmentation layer by twice of the channel size part. For the detailed illustration, see Fig. 3 (b). The affordance map could then be obtained from the second channel of the segmentation output which represents the suctionable class.

4.4 Training models

In this work, we use a PyTorch framework to build, train, and evaluate the models. To train the models, we use a cross entropy loss function to compute the loss value on each training prediction. For the BiSeNet model, we use three different loss values on three different upsample stages to better guide the total output loss, so we can get a more precise weight value, as explained in the original paper[36]. These three different losses will then combined using a weighted sum. For the weight optimizer, we use stochastic gradient descent (SGD) with a fixed momentum value of 0.99 and learning rate of 10^{-3}. This parameter value of the optimizer is chosen as it empirically performs well to optimize the weight, does not take too long to achieve convergence value, and also stable in completing the entire training without having the problem of gradient explosion. The image pixels which are labeled as unlabeled, which are neither suctionable nor unsuctionable, are ignored in the training so they are trained with 0 weight loss in backpropagation. We use the batch size of 2 for the entire training process as the amount of dataset is not large enough. All of the models are trained on a single NVIDIA GTX1 060 6 GB on an Intel Xeon E3.

To prevent overfitting the model, we also perform an image augmentation operations on the input dataset images in real-time as the models are trained and applied in a specified probability value. This image augmentation operations are done using the Imgaug Library[37][2]. There are three different sequences of augmentation operations which are chosen randomly. The first sequence performs a left-right flip, piecewise affine operation, and a perspective transformation. The second sequence is almost the same as the first, with the difference of using up-down flip rather than left-right flip. The last sequence perform a left-right flip operation, followed by dropout operation, then piecewise affine operation, and finally a perspective transformation operation.

[2]https://imgaug.readthedocs.io/en/latest/

4.5 Evaluating performance

To compare the performance of each model, there are three metrics which are considered in this paper: the precision, speed, and size of the model. In normal conditions, to compute the performance of a semantic segmentation task is done by comparing the binary mask of the predicted data to the label and classifying them into four types: 1) True positive (TP) when the prediction is the same as label and is predicted for a given class. 2) True negative (TN) when the prediction is the same as label and is predicted to be not for a given class. 3) False positive (FP) when the prediction is different from label and is predicted for a given class. 4) False negative (FN) when the prediction is different from label and is predicted to be not for a given class. There are several metrics which can be used to measure the performance of a semantic segmentation models, that are not limited to: accuracy, precision, recall (sensitivity).

The problem with measuring performance for the output of this method is that the output affordance map is represented as a probability value. There is a need to choose a threshold value to determine if a prediction is considered to be true or false. We determine the threshold value is 0.99 percentile (top 1%) of the prediction, which will be used to classify the correct prediction. We then compute the precision and recall with the corresponding equations:

$$recall = \frac{TP}{TP + FN} \tag{1}$$

$$precision = \frac{TP}{TP + FP}. \tag{2}$$

5 Experiment result

5.1 Baseline performance

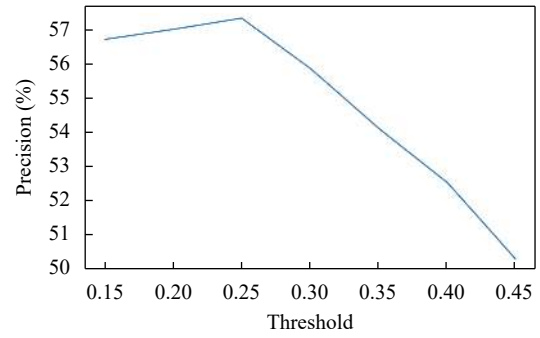
The main concern of the baseline algorithm is to have a general sense of reference about the performance of a "traditional" algorithm which does not have any learning method, so we should expect a better performance with the method that employs a learning method in the solution as is the case with this problem. There are three

variables which could possibly affect the performance of the baseline algorithm: a background subtraction threshold for color image and depth image, and the radius of the local surface normal computation. We can infer that the effect of color image threshold value on precision is that a higher value will generally result in a decrease in precision, with the optimum value as 0.25. It is also true for any different value in the opposing variables which is significantly clearer compared to the effect of depth threshold.

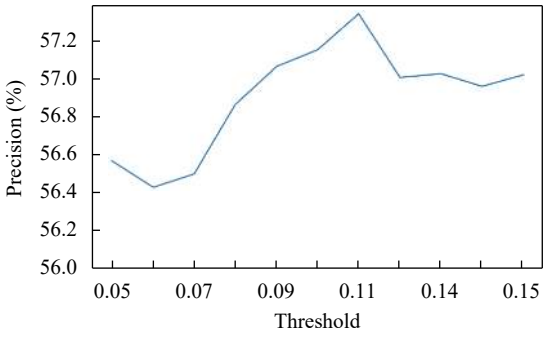
For the effect of depth image threshold value on precision, the pattern is less clear since it fluctuates in some random order, and the pattern of fluctuation also differs with the change of the opposing variables, see Fig. 4 (c). The optimum precision performance that could be achieved using this baseline algorithm for the threshold values are 0.25 for color threshold and 0.11 for depth threshold.

The radius for the local surface normal computation could not be generally considered to have any impact on the precision as it only has a slight dip of precision on radius 0.6 cm to 0.8 cm, and generally remains the same for other values (see Fig. 5), which is possibly caused by a random chance. The larger radius will only increase the precision of the computation, not the quality of the prediction. The algorithm could achieve 57.34% of precision with an average computation time of 803.32 ms for the entire computation of the prediction, while the number of points for surface normal computation is set to 100.

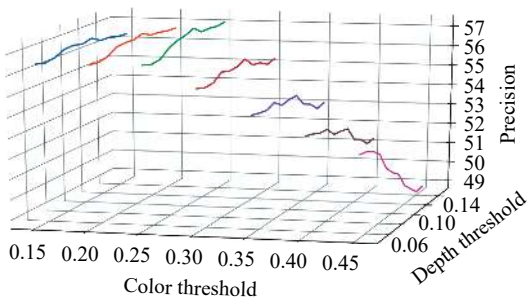
This baseline algorithm could generally achieve a good grasp point performance when predicting a less cluttered scene, but it fails for the highly cluttered scene. This problem is mainly due to the effect of the camera, where the raw pixel value (brightness) could change automatically, especially in scenes where the objects are brighter or darker, which causes a higher difference in raw values even though there are no objects in that particular area of pixels. Such a problem also likely to occur when the scene has a higher number of objects in it.



(a) Effect of threshold value for color imageon precision



(b) Effect of threshold value for depth imageon precision



(c) Complete visualization of threshold value effect on precision

Fig. 4 Effect of threshold value for background subtraction on the precision of baseline algorithm

5.2 Model performance

This evaluations are performed using NVIDIA 1060 6GB GPU, with Intel Xeon E3 and 4GB of RAM on an Ubuntu 16.04 operating system. For the software environment, we use PyTorch version 1.2.0 with NVIDIA driver version 415, CUDA version 10.1, and cuDNN version 7.6.3.

Table 1 shows the model performance evaluation results, where each metrics is explained in 4.5. It should be noted that not several backbone-model combinations are not trained due to the lack of resources.

The results of the evaluation shown in Table 1 indicate the model with the highest precision is the combination of ResNet101+PSPNet but only by a small margin compared to ResNet101+FCN and even with the faster model of ResNet50+BiSeNet. It should be noted that the ResNet101+BiSeNet combination and other higher-sized models were not trained because of the lack of GPU memory resources for the training. We could also conclude that the result of the EfficientNet backbone model

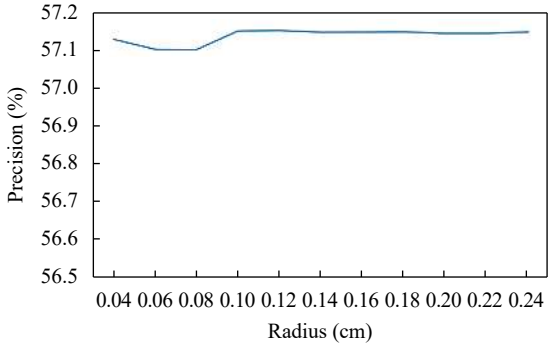


Fig. 5 Effect of local surface normal radius on precision of algorithm

compared to the similar ResNet model size as specified in the original paper[28] is lower as all the segmentation layer models used are designed specifically to have a better suit for ResNet or similar models. Nevertheless, the result of high-sized EfficientNet models could even match the model combination performance of higher equivalent sized models to the ResNet backbone, e.g., EfficientNet-b3+PSPNet which should be equivalent to ResNet50 compared to ResNet101+PSPNet.

The result of evaluation shown in the Table 1 has come as expected where the higher version of the models requires a bigger memory size. It should be noted that the memory required to run a model is not a factor influencing the inference speed of a model. It just corresponds to the amounts of data that are stored in the model which contains the weight, bias, and other parameters required by each layer of the network. For example, EfficientNet backbone models are much more efficient than ResNet models but have a slightly larger memory size. The inference speed of a model is highly influenced by how efficient each layer of the models to perform parallel computation in GPU.

Table 1 Evaluation result of the performance of models

Model		Precision (%)	Time (ms)	Size (MB)
Backbone	Segmentation			
ResNet18	FCN	85.33	11.06	**121.47**
ResNet34		87.36	18.11	248.98
ResNet50		87.91	31.44	292.09
ResNet101		88.27	52.00	727.15
EfficientNet-b0		75.53	19.11	163.01
EfficientNet-b1		76.58	26.13	223.57
EfficientNet-b2		81.57	26.34	251.30
EfficientNet-b3		83.85	29.69	339.97
EfficientNet-b4		86.12	36.02	551.07
ResNet18	PSPNet	87.27	12.64	220.08
ResNet34		88.22	19.09	347.58
ResNet50		88.47	29.78	661.66
ResNet101		**88.83**	53.14	1096.78
EfficientNet-b0		82.27	19.64	628.06
EfficientNet-b1		85.94	27.32	698.24
EfficientNet-b2		86.32	30.23	785.93
EfficientNet-b3		87.25	35.25	961.16
EfficientNet-b4		87.66	36.58	1332.81
ResNet18	BiSeNet	86.54	**6.90**	431.52
ResNet34		87.43	9.81	666.49
ResNet50		88.45	13.10	1230.13
EfficientNet-b0		80.15	20.66	243.88
EfficientNet-b1		84.11	27.95	311.01
EfficientNet-b2		85.40	28.35	369.08
EfficientNet-b3		86.81	31.15	476.42
EfficientNet-b4		87.74	37.42	736.71

In this work, we have also implemented a more streamlined strategy to the post-processing phase by adding a faster and more effective algorithm, and adding several strategies to prevent the model from overfitting compared to the original work by Zeng et al.[4] which is proven to be a better solution. We use the same experimental parameters as Zeng et al.[4], i.e., 20 different objects with 24 suction trials, for fair comparison. In their original work, this part refers to the result of the work from Zeng et al.[4], the top 1% prediction refers to the threshold value to compute precision with ResNet101 backbone and FCN layer, compared to our implementation of the same network which achieves 88.27% of precision. Further, our results with ResNet18+FCN model can achieve a better result compared to those reported in their work as shown in Table 2. Not only that, our implementation also has a significantly lower inference time especially for lower sized models with a higher precision result. It should also be noted that Zeng et al.[4] use NVIDIA Titan X 12 GB for evaluation, which costs 6 times compared to our hardware of NVIDIA GTX 1060 6 GB.

The main drawbacks of our method is that the model does not learn the visual relationships between objects which in some cases play an important role in deciding which object to grasp first before the others. As illustrated in Fig. 6 (b), the model fails to decide the very top object to grasp first but instead it provides a higher affordance value to the object that is stacked by other objects which could possibly lead to a failed grasping attempt if the suction is not strong enough to push away the top object. But generally, our method provides a good grasp proposal even in a cluttered and challenging scene as shown in Fig. 6 (a).

6 Conclusions

A new approach to grasp point estimation was presen-

Table 2 Performance comparison to other works

Method	precision (%)	time (s)
Lenz et al.[38]	N/A*	13.5
Zeng et al.[39]	N/A*	10–15
Dex-Net 2.0[18]	N/A*	0.8
Matsumoto et al.[40]	N/A*	0.20
Zeng et al. (ResNet101+FCN)[4]	83.40	0.060
Ours (ResNet101+FCN)	88.27	0.087
Ours (ResNet18+FCN)	85.33	0.019
Ours (ResNet50+BiSeNet)	88.45	0.03
Ours (ResNet101+PSPNet)	88.83	0.102

*not comparable, using different datasets for evaluation

(a) Good grasp proposal

(b) Failed grasp proposal

Fig. 6 Visualization of the affordance map output of the main model, the left one is the input color image and the right one is the output affordance map.

ted in this work. This approach predicted a probability map which represents the estimation of a success rate for a grasping action in that particular point in 3D space which in this case is represented in the pixel elements of an image input, called affordance map. We use a modified deep neural network models for semantic segmentation tasks to directly compute the affordance map from an RGBD image input of a scene. Two backbone models of ResNet and EfficientNet, and three different semantic segmentation layers that are FCN, PSPNet, and BiSeNet were used. These components of the model are then combined with each other and the performance compared in terms of precision, inference speed, and model size. We achieve the best performance of precision with the combination of the ResNet101+PSPNet model which could achieve a precision of 88.83%. But, in terms of overall performance, the best performing model is the combination ResNet50+BiSeNet model as it could achieve 88.45% with a significantly faster inference speed of 29.8 ms. These results achieved in this work have surpassed the reported performance of the state-of-the-art work by Zeng et al.[4]

References

[1] X. L. Li, L. C. Wu, T. Y. Lan. A 3D-printed robot hand with three linkage-driven underactuated fingers. *International Journal of Automation and Computing*, vol. 15, no. 5, pp. 593–602, 2018. DOI: 10.1007/s11633-018-1125-z.

[2] Y. D. Ku, J. H. Yang, H. Y. Fang, W. Xiao, J. T. Zhuang. Optimization of grasping efficiency of a robot used for sorting construction and demolition waste. *International Journal of Automation and Computing*, vol. 17, no. 5, pp. 691–700, 2020. DOI: 10.1007/s11633-020-1237-0.

[3] C. Ma, H. Qiao, R. Li, X. Q. Li. Flexible robotic grasping strategy with constrained region in environment. *International Journal of Automation and Computing*, vol. 14, no. 5, pp. 552–563, 2017. DOI: 10.1007/s11633-017-1096-5.

[4] A. Zeng, S. R. Song, K. T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. L. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morena, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, A. Rodriguez. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *Proceedings of IEEE International Conference on Robotics and Automation*, Brisbane, Australia, pp. 3750–3757, 2018. DOI: 10.1109/ICRA.2018.8461044.

[5] S. Caldera, A. Rassau, D. Chai. Review of deep learning methods in robotic grasp detection. *Multimodal Technologies and Interaction*, vol. 2, no. 3, Article number 57, 2018. DOI: 10.3390/mti2030057.

[6] D. M. Zhao, S. T. Li. A 3D image processing method for manufacturing process automation. *Computers in Industry*, vol. 56, no. 8–9, pp. 975–985, 2005. DOI: 10.1016/j.compind.2005.05.021.

[7] A. Collet, D. Berenson, S. S. Srinivasa, D. Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. In *Proceedings of IEEE international conference on Robotics and Automation*, Kobe, Japan, pp. 3534–3541, 2009. DOI: 10.1109/ROBOT.2009.5152739.

[8] C. Papazov, S. Haddadin, S. Parusel, K. Krieger, D. Burschka. Rigid 3D geometry matching for grasping of known objects in cluttered scenes. *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 538–553, 2012. DOI: 10.1177/0278364911436019.

[9] M. Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, T. K. Marks, R. Chellappa. Fast object localization and pose es-

timation in heavy clutter for robotic bin picking. *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 951–973, 2012. DOI: 10.1177/0278364911436018.

[10] C. Y. Tsai, K. J. Hsu, H. Nisar. Efficient model-based object pose estimation based on multi-template tracking and PnP algorithms. *Algorithms*, vol. 11, no. 8, Article number 122, 2018. DOI: 10.3390/a11080122.

[11] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour, S. Schaal. Learning of grasp selection based on shape-templates. *Autonomous Robots*, vol. 36, no. 1–2, pp. 51–65, 2014. DOI: 10.1007/s10514-013-9366-8.

[12] M. Gualtieri, A. Ten Pas, K. Saenko, R. Platt. High precision grasp pose detection in dense clutter. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Daejeon, South Korea, pp. 598–605, 2016. DOI: 10.1109/IROS.2016.7759114.

[13] A. ten Pas, M. Gualtieri, K. Saenko, R. Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, vol. 36, no. 13–14, pp. 1455–1473, 2017. DOI: 10.1177/0278364917735594.

[14] J. Wei, H. P. Liu, G. W. Yan, F. C. Sun. Robotic grasping recognition using multi-modal deep extreme learning machine. *Multidimensional Systems and Signal Processing*, vol. 28, no. 3, pp. 817–833, 2017. DOI: 10.1007/s11045-016-0389-0.

[15] D. Guo, F. C. Sun, H. P. Liu, T. Kong, B. Fang, N. Xi. A hybrid deep architecture for robotic grasp detection. In *Proceedings of IEEE International Conference on Robotics and Automation*, Singapore, pp. 1609–1614, 2017. DOI: 10.1109/ICRA.2017.7989191.

[16] J. Bohg, A. Morales, T. Asfour, D. Kragic. Data-driven grasp synthesis-a survey. *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014. DOI: 10.1109/TRO.2013.2289018.

[17] Y. Xiang, T. Schmidt, V. Narayanan, D. Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes, [Online], Available: https://arxiv.org/abs/1711.00199, 2017.

[18] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Y. Liu, J. A. Ojea, K. Goldberg. Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics, [Online], Available: https://arxiv.org/abs/1703.09312, 2017.

[19] J. Mahler, K. Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In *Proceedings of the 1st Annual Conference on Robot Learning*, Mountain View, USA, pp. 515–524, 2017.

[20] T. T. Do, M. Cai, T. Pham, I. Reid. Deep-6DPose: Recovering 6D object pose from a single RGB image, [Online], Available: https://arxiv.org/abs/1802.10367, 2018.

[21] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects, [Online], Available: https://arxiv.org/abs/1809.10790, 2018.

[22] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, K. Goldberg. Segmenting unknown 3D objects from real depth images using mask R-CNN trained on synthetic data. In *Proceedings of International Conference on Robotics and Automation*, IEEE, Montreal, Canada, pp. 7283–7290, 2019. DOI: 10.1109/ICRA.2019.8793744.

[23] A. Saxena, J. Driemeyer, J. Kearns, A. Y. Ng. Robotic grasping of novel objects. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, Vancouver, Canada, pp. 1209–1216, 2006.

[24] E. Johns, S. Leutenegger, A. J. Davison. Deep learning a grasp function for grasping under gripper pose uncertainty. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Daejeon, South Korea, pp. 4461–4468, 2016. DOI: 10.1109/IROS.2016.7759657.

[25] Q. K. Lu, K. Chenna, B. Sundaralingam, T. Hermans. Planning multi-fingered grasps as probabilistic inference in a learned deep network, [Online], Available: https://arxiv.org/abs/1804.03289, 2018.

[26] C. F. Liu, B. Fang, F. C. Sun, X. L. Li, W. B. Huang. Learning to grasp familiar objects based on experience and objects' shape affordance. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2710–2723, 2019. DOI: 10.1109/TSMC.2019.2901955.

[27] K. M. He, X. Y. Zhang, S. Q. Ren, J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA, pp. 770–778, 2016. DOI: 10.1109/CVPR.2016.90.

[28] M. X. Tan, Q. V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks, [Online], Available: https://arxiv.org/abs/1905.11946, 2019.

[29] B. Zoph, V. Vasudevan, J. Shlens, Q. V. Le. Learning transferable architectures for scalable image recognition, [Online], Available: https://arxiv.org/abs/1707.07012, 2017.

[30] M. X. Tan, B. Chen, R. M. Pang, V. Vasudevan, M. Sandler, A. Howard, Q. V. Le. MnasNet: Platform-aware neural architecture search for mobile, [Online], Available: https://arxiv.org/abs/1807.11626, 2018.

[31] M. Sandler, A. Howard, M. L. Zhu, A. Zhmoginov, L. C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Salt Lake City, USA, pp. 4510–4520, 2018. DOI: 10.1109/CVPR.2018.00474.

[32] J. Hu, L. Shen, G. Sun. Squeeze-and-excitation networks. In *Proceedings of IEEE/CVF conference on Computer Vision and Pattern Recognition*, IEEE, Salt Lake City, USA, pp. 7132–7141, 2018. DOI: 10.1109/CVPR.2018.00745.

[33] J. Long, E. Shelhamer, T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Boston, USA, pp. 3431–3440, 2015. DOI: 10.1109/CVPR.2015.7298965.

[34] H. S. Zhao, J. P. Shi, X. J. Qi, X. G. Wang, J. Y. Jia. Pyramid scene parsing network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, USA, pp. 2881–2890, 2017. DOI: 10.1109/CVPR.2017.660.

[35] F. Yu, V. Koltun. Multi-Scale context aggregation by dilated convolutions, [Online], Available: https://arxiv.org/abs/1511.07122, 2015.

[36] C. Q. Yu, J. B. Wang, C. Peng, C. X. Gao, G. Yu, N. Sang.

BiSeNet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the 15th European Conference on Computer Vision*, Springer, Munich, Germany, pp. 334–349, 2018. DOI: 10.1007/978-3-030-01261-8_20.

[37] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernández, W. F. M. De Rainville, Chi-Hung, A. Ayala-Acevedo, R. Meudec, M. Laporte. Imgaug, [Online], Available: https://github.com/aleju/imgaug, May 5, 2019.

[38] I. Lenz, H. Lee, A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, vol. 34, no. 4–5, pp. 705–724, 2015. DOI: 10.1177/0278364914549607.

[39] A. Zeng, K. T. Yu, S. R. Song, D. Suo, E. Walker Jr, A. Rodriguez, J. X. Xiao. Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge, [Online], Available: https://arxiv.org/abs/1609.09475, 2016.

[40] E. Matsumoto, M. Saito, A. Kume, J. Tan. End-to-end learning of object grasp poses in the amazon robotics challenge. *Advances on Robotic Item Picking*, A. Causo, J. Durham, K. Hauser, K, Okada, A. Rodriguez, Eds., Cham, Switzerland: Springer, pp. 63–72, 2020. DOI: 10.1007/978-3-030-35679-8_6.

Tri Wahyu Utomo received the B. Eng. degree in electrical engineering from University of Gadjah Mada, Indonesia in 2019. He is now working as an artificial intelligence (AI) engineer in a computer vision startup company in Jakarta, Indonesia.

His research interests include motion planning for autonomous mobile robot, deep learning and computer vision.

E-mail: tri.wahyu.u@mail.ugm.ac.id
ORCID iD: 0000-0002-0551-5594

Adha Imam Cahyadi received the B. Eng. degree in electrical engineering from University of Gadjah Mada, Indonesia in 2002. Then he worked as an engineer in industry, such as in Matsushita Kotobuki Electronics and Halliburton Energy Services for a year. He received the M. Eng. degree in control engineering from King Mongkut's Institute of Technology Ladkrabang, Thailand (KMITL) in 2005, and received the Ph. D. degree in control engineering from Tokai University, Japan in 2008. Currently, he is a lecturer at Department of Electrical Engineering and Information Technology, University of Gadjah Mada and a visiting lecturer at the Centre for Artificial Intelligence and Robotics (CAIRO), University of Teknologi Malaysia, Malaysia.

His research interests include teleoperation systems and robust control for delayed systems especially process plant.

E-mail: adha.imam@ugm.ac.id
ORCID iD: 0000-0002-9825-2630

Igi Ardiyanto received the B. Eng. degree in electrical engineering from University of Gadjah Mada, Indonesia in 2009, the M. Eng. and Ph. D. degrees in computer science and engineering from Toyohashi University of Technology (TUT), Japan in 2012 and 2015, respectively. He joined the TUT-NEDO (New Energy and Industrial Technology Development Organization, Japan) research collaboration on service robots, in 2011. He is now an assistant professor at University of Gadjah Mada, Indonesia. He received several awards, including Finalist of the Best Service Robotics Paper Award at the 2013 *IEEE International Conference on Robotics and Automation* (ICRA 2013) and Panasonic Award for the 2012 RT-Middleware Contest.

His research interests include planning and control system for mobile robotics, deep learning, and computer vision.

E-mail: igi@ugm.ac.id (Corresponding author)
ORCID iD: 0000-0002-0006-1458