DSC-GraspNet: A Lightweight Convolutional Neural Network for Robotic Grasp Detection

Zhiyang Zhou
School of Information Engineering
Southwest University of Science and Technology
Mianyang, China
zhouzhiyang@swust.mails.edu.cn

Xiaoqiang Zhang*
School of Information Engineering
Southwest University of Science and Technology
Mianyang, China
xqzhang@swust.edu.cn

Lingyan Ran
School of Computer Science
Northwestern Polytechnical University
Xi'an, China
lran@nwpu.edu.cn

Yamin Han
College of Information Engineering
Northwest A&F University
Xianyang, China
yaminhan@nwafu.edu.cn

Hongyu Chu
School of Information Engineering
Southwest University of Science and Technology
Mianyang, China
chuhongyu@swust.edu.cn

2023 9th International Conference on Virtual Reality (ICVR) | 979-8-3503-4581-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICVR57957.2023.10169448

***Abstract*—Grasp detection is an essential task for robots to achieve autonomous operation, it can also make virtual reality-based teleoperation more intelligent and reliable. Existing learning-based grasp detection methods usually fail to strike a balance between high accuracy and low time consumption. Also, the large number of model parameters tends to make these methods expensive to deploy. To solve this problem, a lightweight generative grasp detection network DSC-GraspNet is proposed. Firstly, Depth-separable convolutional blocks with Coordinate Attention (CA) are stacked to obtain a lightweight backbone network for feature extraction. Then multi-level features extracted by the backbone network are fused by the Cross Stage Partial (CSP) block in the up-sampling network. Finally, pixel-level grasp candidates are generated by grasp generating heads. Experimental results shows that an accuracy of 98.3% under image-wise splitting and 97.7% under object-wise splitting can be achieved on the Cornell public dataset. Meanwhile, an accuracy of 94.7% is achieved on the Jacquard dataset using the depth map as inputs. Our method also achieve a grasp success rate of 86.4% in the simulated grasp test. In addition, our network is able to inference an RGB-D image within 14ms, and can be applied to closed-loop grasping scenarios.**

***Index Terms*—Robot, grasp detection, convolutional neural network, depth-separable convolution.**

I. Introduction

Grasp detection is a key technology for robot autonomous operations, and it is also widely used in the field of robot teleoperation based on virtual reality (VR) [1], [2]. Traditional grasp detection methods [3], [4] rely too much on manually designed features, which is tedious and time-consuming. With the increase in computer performance, the performance of deep learning methods on the problem of grasp detection has improved greatly. These deep learning-based grasp detection methods can be broadly classified into three categories: classification-based, regression-based methods and generation-based methods. Classification-based methods [5]–[7] usually use a sliding window to traverse the entire image to find the optimal grasp candidate, which requires a lot of time and computational resources. Regression-based methods [8]–[10] directly regress the grasp pose by CNN, which performs well in practical grasping. However, the regression-based methods also suffers from the problems of large model size, long inference time for a single image, and high network deployment cost. To overcome the above problems, generation-based grasp detection methods based on CNN is proposed [11]–[13]. Different from previous methods, generation-based methods generate pixel-wise grasp poses based on the global information of the image, similar to image segmentation tasks. Generation-based grasp detection method does not require a large network to extract image features for grasp pose prediction, so the amount of parameters is usually much smaller than the previous two types of methods, and can achieve high computational efficiency.

In this work, a generation-based **D**epth-**S**eparable **C**onvolution-based grasp detection network (DSC-GraspNet) that generates pixel-level grasp candidates is proposed. DSC-GraspNet consists of three parts: the backbone network, the up-sampling network, and the grasp generating head. In order to reduce the size of the network, the DepthSepConv block in MobileNet V1 [14] are introduced into the backbone network. Subsequently, we add the Coordinate Attention (CA) [15] to the DepthSepConv block to further improve the feature extraction capability of the network. Moreover, by integrating the Cross-Stage Partial (CSP) block with bottleneck layer, the up-sampling network can fuse the multi-layer features from the backbone network, which would improves the feature utilization and ensures the speed of inference. The contributions of this paper can be summarised in three folds:

- A lightweight generative grasp detection network architecture is proposed that predicts the optimal grasp configuration for objects in n-channel images. The network also has high detection accuracy while keeping the number of parameters low.
- A series of quantitative comparative experiments on pub-

*Xiaoqiang Zhang is the corresponding author.

licly available datasets are conducted, and our network achieving 98.3% and 94.7% accuracy on the Cornell and Jacquard grasp datasets, respectively.
- Robot grasping experiments in a simulated environment based our method are performed. The 7 DOF robot make more than 150 grasping attempts on different unknown objects, with a success rate of 86.4%.

II. Related work

In the field of robotics, the grasp detection has been a widely and continuously studied problem. In the last decade, with the improvement of computer performance, deep learning-based method have also been developed and widely used in the field of grasp detection, which can be classified into three categories, namely, classification-based method, regression-based method and generation-based method.

A. Classification-based method

Classification-based method use classifiers to select the highest scoring grasps. For example, Lenz *et al*. [5] first propose a grasp detection network trained by sparse self-encoders using sliding windows, finally the network achieve a detection accuracy of 75.6% and an inference time of 1350 ms on the Cornell dataset. Wang *et al*. [6] propose a convolutional neural network-based classification network to identify candidate grasping regions. Pinto *et al*. [7] use a CNN network to predict the grasp position and angle of image sample patch with better generalization capability in the face of unknown objects. These classification-based methods show relatively high accuracy in grasp detection, but still fall short of fast inference and low computational consumption.

B. Regression-based method

Regression-based methods use CNN to directly regress the grasping position and angle. In early works, Redmon *et al*. [8] propose a large network based on a single-stage regression approach to predict grasps. Kumra *et al*. [9] design a CNN using ResNet [16] as the feature extractor to predict grasps in multi-modal data inputs, while they achieve an accuracy of 89.2% and an inference time of 103 ms on the Cornell dataset. To improve the feature representation, Zhang *et al*. [10] construct a multi-modal fusion architecture using dilated convolution and a novel loss function. These regression-based methods perform well in grasp detection, but suffer from the drawbacks of large network size and long inference time.

C. Generation-based method

Unlike the previous two types of methods, the generation-based methods generates pixel-level grasp candidates directly. Inspired by the fully convolutional network FCN [17], GG-CNN algorithm is proposed by Morrison *et al*. [11], which uses the pixel-level labeled grasp map to complete the grasp detection task. As the first generation-based grasp detection method, GG-CNN achieved the grasp detection accuracy of 73.0% and the fastest inference speed of 19 ms on the Cornell dataset. Wang *et al*. [18] improve GG-CNN and further use the global and local features of the image to improve the grasp detection accuracy. Kumra *et al*. [12] propose GR-ConvNet, which introduce multiple residual structures to obtain higher accuracy. Multi-task learning is introduced into the grasp detection network by Prew *et al*. [13]. Wang *et al*. [19] present a new transformer-based architecture TF-Grasp for robotic grasp detection, which obtain a state-of-the-art performance in public datasets.

III. Method

In this section, the proposed method will be described in detail. The grasp representation we used is introduced in subsection III-A. In subsection III-B to III-E, each key part of the grasp detection network are presented in detail. The loss function used to optimize the network during the training process is described in subsection III-F.

A. Grasp representation

Consistent with previous work [12], [13], [19], an improved version of grasp representation proposed by Morrison *et al*. [11] is used in this paper:

$$G_i = (x, y, \Theta_i, W_i, Q), \tag{1}$$

where (x, y) represents the center point of grasp in image frame, Θ_i the orientation in the camera frame, W_i the required width in image frame, and Q the grasp quality score. The above grasp is defined in the 2D image frame, and we need to convert it to the robot reference system, this process that can be described as:

$$G_r = T_{rc}(T_{ci}(G_i)), \tag{2}$$

where T_{rc} and T_{ci} are the transform matrix of the camera frame to the robot frame and 2D image frame to the camera frame, respectively. This notation can be scaled for multiple grasps in an image. The set of all the grasps can be denoted as:

$$\mathbf{G} = (\mathbf{\Theta}, \mathbf{W}, \mathbf{Q}) \in \mathbb{R}^{3\times h\times w}, \tag{3}$$

where $\mathbf{\Theta}, \mathbf{W}$, and $\mathbf{Q}$ represents three images in the form of grasp angle, grasp width and grasp quality score calculated at every pixel of an image using Eq. (1), respectively. Finally, the center point's position can be calculated by searching for the pixel value of the maximum grasp quality: $\mathring{g} = \max_{\mathbf{Q}}(\mathbf{G})$.

B. Network architecture

Inspired by GG-CNN [11], a Depth-Separable Convolution-based generative grasping detection network DSC-GraspNet is proposed. Different from GG-CNN, n-channel data is used as network input to obtain richer feature expressions. In addition, the feature extraction network and upsampling network have also been improved. Fig. 1 demonstrates the network architecture. The network takes n-channel images as input, and the end-to-end output contains a grasping set $\mathbf{G}$ of grasp quality, grasp angle and grasp width. DSC-GraspNet is divided into three parts: the backbone network, the up-sampling network and the grasp generating head. We use the backbone network

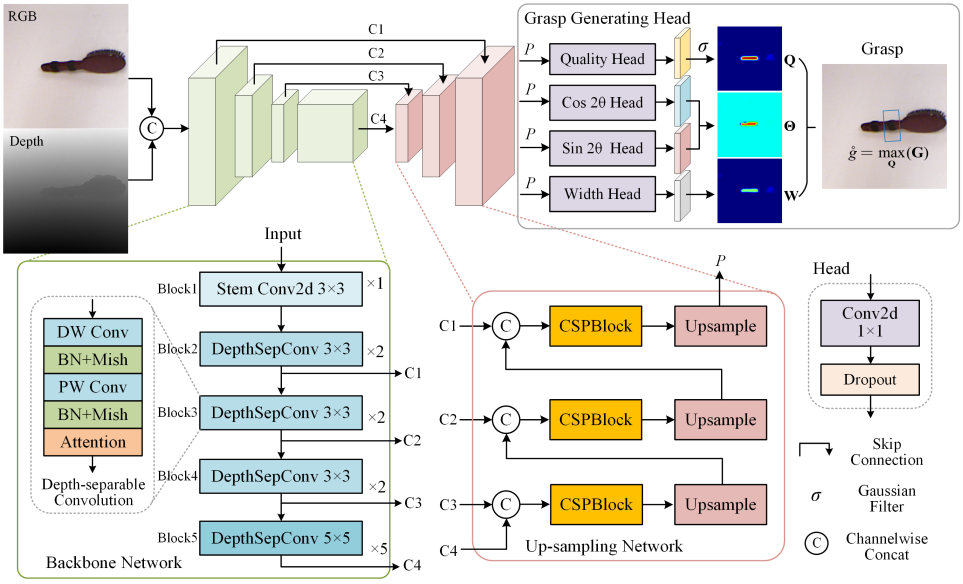

Fig. 1. DSC-GraspNet Architecture. The backbone network is staked by the improved DepthSepConv blocks, which takes the n-channel image **I** as the input, and outputs $C1 - C4$ feature maps to the up-sampling network. The Up-sampling Network take the input from the $C1 - C4$ feature maps and outputs one feature map **P** to the grasp generating head. The head finally generate the grasp pose of each pixel (the grasp map **G**) by four heads, including the grasp quality **Q**, grasp width **W**, and grasp angle **Θ**. Finally, the optimal grasp pose can be calculated by $\mathring{\boldsymbol{g}}_{\boldsymbol{i}} = \max_{\mathbf{Q}}(\mathbf{G})$.

to extract multi-level semantic information of the input image and obtain feature maps, which will be sent to the up-sampling network. The up-sampling network will fuse and upsample these feature maps and restore the size of the feature maps to the same size as the input image. Finally, the grasp quality **Q**, grasp angle **Θ** and the opening width of the gripper **W** are obtained by the grasp generating heads.

C. Backbone network

Most of the current generative grasp detection networks use fully convolutional neural network architectures, which use a conventional convolutional layer superposition structure when performing feature extraction. For example, the GG-CNN network uses basic convolution and maximum pooling to form the feature extraction network, and upsamples the feature map by deconvolution to finally generate the grasp map. While deepening the depth of the convolutional neural network or widening the width of the network would improve the data fitting ability and the prediction accuracy of the network, the increase of the amount of parameters would result in an increase of inference time and deployment cost, which restrict the real-time application in grasp detection networks. To address this problem, we propose a light-weight backbone network for feature extraction based on depth-separable convolution and coordinate attention is utilized in the proposed network.

1) Depth Separable Convolution Block: the depth-separable convolution was first consists of two processes: depth-wise convolution (DWConv) and point-wise convolution (PWConv). The number of parameters in depth-separable convolution is about one-third of that in traditional convolution, so the number of layers of the network can be deeper when we stack them using depth-separable convolution with the same amount of network parameters. We use the DepthSepConv

TABLE I
THE DETAILS OF THE BACKBONE NETWORK

Operator	Size	Stride	Input	Output	CA
Stem / Conv2d	3×3	1	$224^2 \times 4$	$224^2 \times 32$	–
DepthSepConv	3×3	1	$224^2 \times 32$	$224^2 \times 32$	√
DepthSepConv	3×3	2	$224^2 \times 32$	$112^2 \times 64$	√
DepthSepConv	3×3	1	$112^2 \times 64$	$112^2 \times 64$	√
DepthSepConv	3×3	2	$112^2 \times 64$	$56^2 \times 128$	√
DepthSepConv	3×3	1	$56^2 \times 128$	$56^2 \times 128$	√
DepthSepConv	3×3	2	$56^2 \times 128$	$28^2 \times 256$	√
5×DepthSepConv	5×5	1	$28^2 \times 256$	$28^2 \times 256$	√

mentioned by MobileNetV1 [14] as basic block, and the structure of DepthSepConv is given in Fig 1. Given the input features, i.e., $F_{in} \in R^{C \times H \times W}$, we first extract the feature by DWConv, and then activate it using Batch Normalization and Mish function to obtain the feature $F_{DCBM} \in R^{C \times H \times W}$, followed by convolution of F_{DCBM} using PWConv, in which we also use the batch normalization and mish function. During experiments, we find that using the Mish function makes the training process more stable. In addition, we also use the attention module after the PWConv layer and get the output feature F_{DSC} for DepthSepConv block. We can formulate the above procedure as Eq. (4):

$$\begin{aligned} F_{DCBM} &= Mish\left(BN\left(DWConv\left(F_{in}\right)\right)\right), \\ F_{PCBM} &= Mish\left(BN\left(PWConv\left(F_{DCBM}\right)\right)\right), \\ F_{DSC} &= Attention\left(F_{PCBM}\right). \end{aligned} \tag{4}$$

Different size of convolution kernels are utilized for the backbone network, and the specific structure of backbone is shown in Table. I.

2) Coordinate Attention: Most existing attention to channel operations use maximum pooling or average pooling, which may lead to loss of spatial information of the target. Therefore, we adopt a new Coordinate Attention (CA) mechanism [15], as shown in Fig. 2. Unlike channel attention that converts

the feature tensor into a single feature vector through 2D global pooling, the CA decomposes channel attention into two 1D feature encoding processes that aggregate features along 2 spatial directions, respectively. In this way, remote dependencies can be captured along one spatial direction, while accurate location information can be retained along the other spatial direction. The generated feature maps are then encoded as a pair of direction-aware and location-sensitive feature vectors, respectively, which can be complementarily applied to the input feature maps by tensor multiplication to weaken redundant features while enhancing the features of interest.

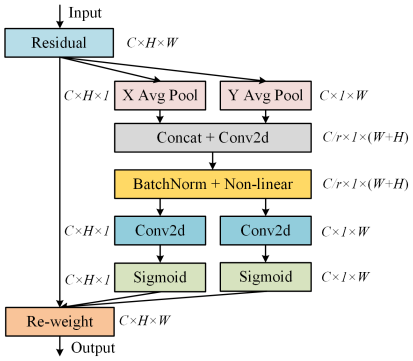

Fig. 2. Schematic of the Coordinate Attention (CA) block [15]. 'X Avg Pool' and 'Y Avg Pool' refer to 1D horizontal global pooling and 1D vertical global pooling, respectively. And the scaling factor r of CA block we set to 32.

D. CSP block based up-sampling network

Different from the backbone network, the up-sampling network focuses more on how to improve the efficiency of feature utilization while ensuring low computational effort. We use the CSP structure in the cross-stage partial network (CSPNet) [20] in each stage of the up-sampling network, and further extract features through the bottleneck in the CSP structure to improve the detection accuracy without affecting the network operation speed, thus solving the problem of imbalance between accuracy and inference speed. More specifically, our up-sampling network is divided into 3 up-sampling stages, each stage consists of a CSP block and a bilinear up-sampling layer. Through 3 stages of up-sampling, we recover the size of the feature map to the size of the original input image. To improve the feature utilization, we use the skip connection structure. We first upsample the output features of the CSP block by bilinear interpolation to obtain the F_{up}, and then concatenate the F_{up} and the feature c_i corresponding to the output of the backbone network in channel dimension. This process can be expressed as Eq. (5):

$$\begin{aligned} F_{up}(x) &= Upsample(x), \\ F_{csp}(x_1, x_2) &= Mish(CSP(Concat(x_1, F_{up}(x_2)))). \end{aligned} \tag{5}$$

A total of three up-sampling stages to recover the size of the feature map was used, and finally we obtain the feature F with the same size as the input image, and the process of up-sampling network can be expressed by Eq. (6):

$$P = F_{csp}^{2}\left(C_1, F_{csp}^{1}\left(C_2, F_{csp}^{0}(C_3, C_4)\right)\right). \tag{6}$$

As shown in Fig. 3 (a), the CSP block inputs the original feature map into two branches. The Path1 branch is composed of convolution, BathNormal and Mish activation function (CBM) in series, and the number of output channels is reduced by half through convolution. The Path2 branch also reduces the number of channels through CBM, and then feeds the features into N bottleneck blocks. Finally, the output of the two branches is concatenated in channel-wise, and then a convolution is performed again through a CBM block. In this way, when the model weight is updated, the combination of gradients is more diverse and the learning ability of the network is improved. At the same time, CSP blocks can also alleviate gradient information redundancy, reduce computing bottlenecks, and reduce memory cost. The structure of the bottleneck is shown in Fig. 3 (b). The first CBM block uses 1×1 convolution to adjust the number of feature channels, and then recovers the channels through 3×3 convolution. In addition, we also increase the gradient value of back propagation between layers by adding residual connection, further avoiding the problem of gradient disappearance, so that the network can learn more refined features.

E. Grasp generating head

The grasp generating head consists of 4 task-specific convolutional layers with a kernel size of 3×3: the *Quality Head*, the *Cos2θ Head*, the *Sin2θ Head*, and the *Width Head*. Experiments show that the network has the potential to overfit during training, so we add a dropout layer for regularization after each header output, which facilitates the network to learn rare but useful features. We use a Gaussian kernel to filter the grasp quality images, which can make the grasp results more robust by removing the extreme values close to the regions with poor grasp quality. The final grasp position (x, y) is the position with the max grasp quality in the quality map. Based on the same position, the angle Θ and the width W can be got in the other images. The rotation angle of the gripper along the normal of the grasp plane can be calculated by $\theta = \frac{1}{2}\arctan\frac{\sin 2\theta}{\cos 2\theta}$.

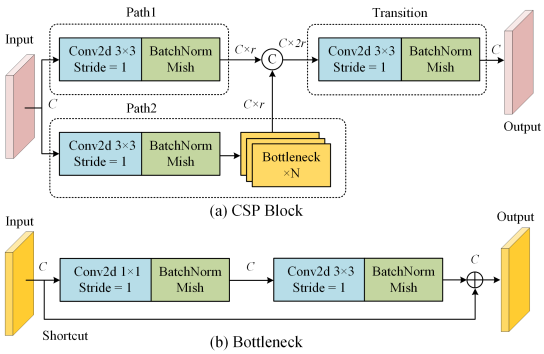

Fig. 3. The CSP Block and bottleneck used by the up-sampling network. (a) The framwork of CSP Block. (b) The bottleneck module. The input features are divided into two parts and sent to Path1 and Path2. Each path adjusts the number of channels through 2d convolution. In Path2, N bottleneck modules are stacked in series, and the feature maps of the two paths are concatenated and feed into the trasition layer.

F. Loss function

The generative grasp detection task is a typical regression task. The Huber function is utilized as the loss function of DSC-GraspNet. The Huber function has the advantages of MSE and MAE function, which is more stable for outliers and has strong robustness in the training process. If direct use Huber function to calculate the regression loss of grasp angle or width images, the network would tend to learn grasp angle or width values close to 0 in non-grasping regions, even if no grasping is attempted in these regions. To avoid this problem, a new position-enhanced loss function P-Huber was proposed. The P-Huber loss function is defined as:

$$L_p = \mathcal{L}\left(Q_e\right) + \mathcal{L}\left(\widehat{Q_i} \cdot \Phi_e^{\cos}\right) + \mathcal{L}\left(\widehat{Q_i} \cdot \Phi_e^{\sin}\right) + \mathcal{L}\left(\widehat{Q_i} \cdot W_e\right), \tag{7}$$

where $\mathcal{L}$ is the Huber function, $Q_e, \Phi_e^{\cos}, \Phi_e^{\sin}$ and W_e the error of quality, Cos2θ, Sin2θ and width of grasp, respectively, and $\widehat{Q_i}$ the Ground Truth grasp quality. The network will more focused on the learning of grasp angle and grasp width at potential grasp region by multiplying error directly with $\widehat{Q_i}$.

IV. Experimental results

In this section, experimental results would be presented to demonstrate the effectiveness of our method. Firstly, grasp datasets and metrics are introduced. Then, quantitative performance comparison experiments are carried out in detail on two public grasp datasets. Finally, the performance of our method in robot grasping is verified by simulation experiments.

A. Datasets and metrics

1) Datasets: The Cornell dataset [21] and the Jacquard dataset [22] are used to validate the effectiveness of the proposed method, which contain both RGB and depth images. The Cornell dataset consists of 885 RGB-D images of 240 different objects. The dataset is manually labeled with 5110 positive and 2090 negative grasps, and is the most widely used benchmark in the field of robotic grasp detection. The Jacqurad dataset is a large grasping dataset created by simulation based on the CAD model of objects, containing 50k RGB-D images of 11k objects and over 1M grasp labels.

2) Metrics: To ensure the fairness of the comparison, we adopt the metric [21] proposed by Jiang *et al.* . This metric states that a candidate grasp is considered correct if it satisfies both of the following conditions:

- **Angle difference:** The difference in orientation Angle between the generated grasp and the grasping label is less than 30° .
- **Jaccard index:** The Jaccard index of the generated grasp and the corresponding ground truth is greater than 25%, which can be formulated as:

$$J(g_p, g_l) = \frac{|g_p \cap g_l|}{|g_p \cup g_l|}, \tag{8}$$

where g_p and g_l denote the generated grasp rectangle and the ground truth grasp label, respectively. $g_p \cap g_l$ represents the intersection of predicted grasp and the ground truth grasp label. And the union of predicted grasp and the corresponding ground truth is represented as $g_p \cup g_l$.

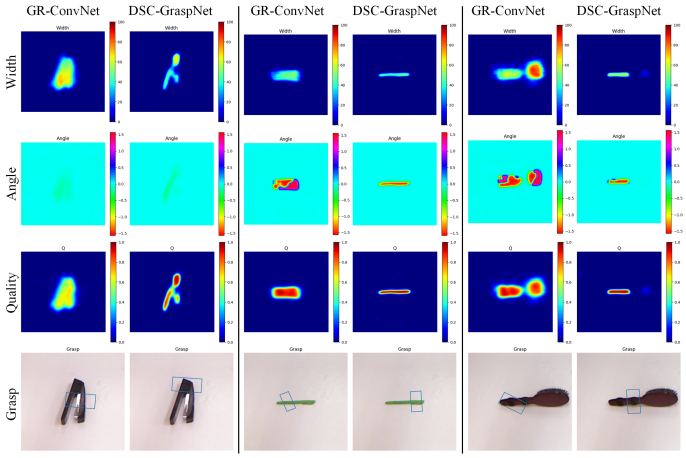

Fig. 4. Comparison studies on the Cornell dataset. The first three rows are the maps of grasp width, angle and the quality generated by two networks. And, the last row is the visualization of the optimal grasp for each objects.

3) Data preprocessing: Due to the relatively small amount of data in the Cornell dataset, we performed data augmentation during the training process. Specifically, we first crop the image based on the center of the original image, and rotate the cropped image randomly, the cropped image is randomly rotated $\pm\frac{\pi}{2}$ or $\pm\pi$. After the rotation, the image is randomly reserved around the center by 50% to 100%, and then resized to the original size. At the same time, the grasp labels are processed accordingly. As Jacquard dataset is large enough to train the network, no data augmentation is performed.

4) Training Methodology: In the training and testing period, the network is given an input image of size 224×224. The hardware system consists of an AMD Ryzen7-5800H CPU @3.20GHz × 8 processors, and an NVIDIA GeForce RTX 3070 Laptop GPU with 8GB memory. The training and inference framework are built based on Pytorch 1.8.2 with cuda-11.0.5 and cudnn-8.0.5 pacakges. The newly proposed Ranger optimizer [23] is adopted to train the model. In addition, we fixed the learning rate 0.001 during training, and the batch size is 32 and the epoch number is 50.

B. Experiments on the Cornell dataset

Following the previous works [5], [8], [9], [24], [25], the Cornell dataset can be divided into two different ways to validate the generalization ability of the method, namely,

- **Image-wise level:** the images of dataset are randomly divided by shuffling. This method is used to evaluate the generalization ability of the network on orientation change and size variation of objects.
- **Object-wise level:** the object instances of dataset are randomly divided. This method is used to evaluate the generalization ability of the network for new object.

The comparison of the accuracy of our DSC-GraspNet with other methods for grasp detection on the Cornell dataset is reported in Table. II. Experimental results show that our DSC-GraspNet achieves high accuracy of 98.3% and 97.7% in IW and OW split with the lowest inference time: 14ms. Not only

TABLE II
QUANTITATIVE COMPARISON RESULTS ON THE CORNELL DATASET

Method	Accuracy (%) IW	OW	Speed (ms)	Parameters (million)
SAE [5]	73.9	75.6	1350	-
AlexNet, MultiGrasp [8]	88.0	87.1	76	-
Two-stage closed-loop [6]	85.3	-	140	-
ResNet-50x2 [9]	89.2	88.9	103	-
GG-CNN [11]	73.0	69.0	19	**0.066**
Multi grasp, ResNet-50 [26]	96.0	96.1	120	216
FCGN, ResNet-101 [27]	97.7	96.6	117	-
GRPN [28]	88.7	-	200	-
STEM-CaRFs [29]	88.2	87.5	-	-
GR-ConvNet [12]	97.7	96.6	20	1.9
TF-Grasp [19]	97.9	96.7	41.6	5.8
Ours DSC-GraspNet-D	96.0	96.6	**11**	
Ours DSC-GraspNet-RGB	97.7	**98.3**	13	0.643
Ours DSC-GraspNet-RGB-D	**98.3**	97.7	14	

TABLE III
QUANTITATIVE COMPARISON RESULTS ON THE JACQUARD DATASET

Method	Accuracy (%)
Jacquard [22]	74.2
GG-CNN [11]	84
FCGN, ResNet-101 [27]	91.8
GR-ConvNet [12]	94.6
TF-Grasp [19]	94.6
Ours DSC-GraspNet-RGB-D	93.8
Ours DSC-GraspNet-D	**94.7**
Ours DSC-GraspNet-RGB	91.8

do we achieve the highest accuracy with RGB-D data input, but our accuracy remains impressive with single RGB or Depth image data input. The parameter amount of our model is about 0.643 million, which is much smaller than most other algorithms, and we have a faster inference speed.

Some results of the grasp detection on the Cornell dataset are visualized in Fig. 4. Note that these images are all new objects that have not appeared in the training set. We use the current state-of-the-art GR-ConvNet [12] as a comparison, and for a fair comparison, we use its officially provided code to train and test on the same dataset. Only the grasp with the highest grasp quality is selected, and the optimal grasp is visualised through blue rectangular box in the last row. It can be seen from the figure that our DSC-GraspNet can generate reliable grasps for unseen objects with different shapes and poses.

C. Experiments on the Jacquard dataset

To further evaluate the performance of the proposed method, experiments will also be conducted on the Jacquard dataset. We follow the setup of existing work and train DSC-GraspNet on 90% of the dataset and validate it on 10% of the remaining dataset. Similar to that of the Cornell dataset, we conducted experiments using multiple input data patterns, and the relevant results are reported in Table. III. Using depth data as input, our DSC-GraspNet achieves performance with 94.7% detection accuracy, outperforming existing methods. We compare DSC-GraspNet with GR-ConvNet on the Jacquard dataset, and the relevant comparison results are visualized in Fig. 5.

D. Ablation study

To further explore the effect of different components on the learning of grasp poses, we trained our models with different

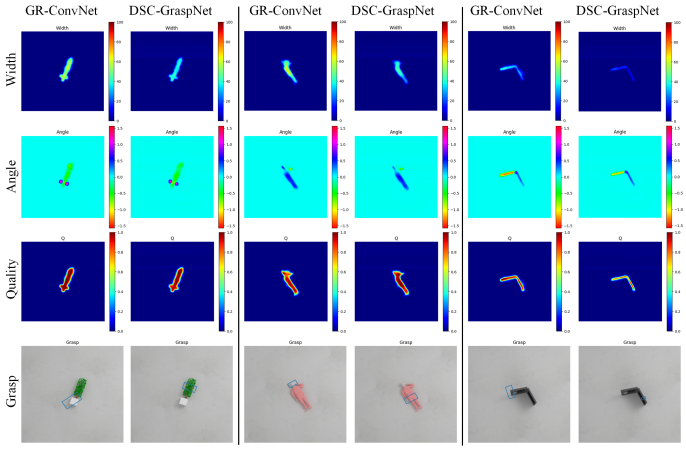

Fig. 5. Comparison studies on the Jacquard dataset. The first three rows are the maps of grasp width, angle and the quality generated by two networks. And, the last row is the visualization of the optimal grasp for each objects.

TABLE IV
DIFFERENT CONFIGURATIONS OF ABLATION EXPERIMENTS IN DSC-GRASPNET

Baseline	✓	✓	✓	✓
+ CA		✓	✓	✓
+ CSP	✓		✓	✓
+ P-Huber	✓	✓		✓
Acurracy (%)	93.3%	93.5%	94.3%	**94.7%**

network settings on the Jacquard dataset with depth images as inputs. The experimental results are summarized in Table. IV. We will include only the backbone and a network consisting of basic convolution and up-sampling layers as the baseline. From the detection accuracy in the Table. IV, we can see that adding CA, CSP and P-Huber loss functions can improve the performance of the network, and in addition we can achieve the best detection accuracy by combining all components together.

E. Failure cases analysis

During the evaluation experiments conducted, there were some cases of failed detection, and some failure cases are shown in Fig. 6. For some objects with complex textures in the Cornell dataset our model does not detect them very well. In addition, the detection quality of some objects with similar color and flatter background in the Jacquard dataset is also poor. These shortcomings can be solved by increasing the diversity of the dataset.

F. Experimental results of simulated grasping

To evaluate antipodal robotic grasping in simulation, we developed a simulation environment in PyBullet [30], where a 7 DOF Panda arm from Franka Emika with a parallel-jaw gripper. An build-in RGB-D visual camera is used to observe the robot's workspace and sense the environment. The objects

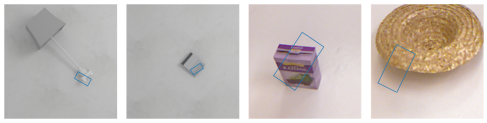

Fig. 6. Some failed detection cases: left two results from Jacquard dataset and right two from Cornell dataset, objects are all unseen in training set.

to be grasped in the experiments are all from the YCB object set. At the beginning of each grasping experiment, the robot will be set to a predetermined pose, with randomly selected objects placed in any position in the robot's workspace. In the experimental process, we predict the optimal grasping pose through the network and send it to the controller of the robot. After the robot moves to the grasping position, it closes the gripper and picks up the object, if the object does not fall, the grasp is considered as successful. For the same object we performed 20 randomized grasping experiments for more than 150 grasping attempts. Our DSC-GraspNet achieves a success rate of 86.4%, while the comparison algorithm GR-ConvNet [12] only achieves a success rate of 77.5%.

V. Conclusion

In this paper, a lightweight generative grasp detection network was proposed, which uses n-channel of input images to generate pixel-level grasp predictions. Quantitative evaluation experiments on the Cornell and Jacquard dataset demonstrate the effectiveness of our method. In addition, the result of simulation grasp experiments show that our method has better performance in terms of accuracy and robustness. Moreover, the low inference time of our model enables its application to closed-loop robotic grasping.

Acknowledgment

This work is supported in part by the Natural Science Foundation of China under Grant 62201479, in part by the Natural Science Foundation of Sichuan Province under Grant 2023NSFSC1388, in part by the Doctoral Fund of Southwest University of Science and Technology under Grant 19zx7123, in part by the Open Fund of Key Laboratory of Civil Aircraft Airworthiness Technology under Grant SH2020112706.

References

[1] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5628–5635.

[2] J. Aleotti and S. Caselli, "Grasp recognition in virtual reality for robot pregrasp planning by demonstration," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2801–2806.

[3] A. Ramisa, G. Alenya, F. Moreno-Noguer, and C. Torras, "Learning rgb-d descriptors of garment parts for informed robot grasping," *Engineering Applications of Artificial Intelligence*, vol. 35, pp. 246–258, 2014.

[4] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single image 3d object detection and pose estimation for grasping," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3936–3943.

[5] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[6] 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.

[7] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 3406–3413.

[8] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1316–1322.

[9] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 769–776.

[10] H. Zhang, X. Zhou, X. Lan, J. Li, Z. Tian, and N. Zheng, "A real-time robotic grasping approach with oriented anchor box," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[11] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *The International Journal of Robotics Research*, p. 0278364919859066, 2019.

[12] S. Kumra, S. Joshi, and F. Sahin, "Antipodal robotic grasping using generative residual convolutional neural network," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9626–9633.

[13] W. Prew, T. Breckon, M. Bordewich, and U. Beierholm, "Improving robotic grasping on monocular images via multi-task learning and positional loss," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 9843–9850.

[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[15] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13 713–13 722.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer, 2016, pp. 630–645.

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[18] S. Wang, X. Jiang, J. Zhao, X. Wang, W. Zhou, and Y. Liu, "Efficient fully convolution neural network for generating pixel wise robotic grasps with high resolution images," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 474–480.

[19] S. Wang, Z. Zhou, and Z. Kan, "When transformer meets robotic grasping: Exploits context for efficient grasp detection," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8170–8177, 2022.

[20] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[21] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3304–3311.

[22] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[23] L. Wright and N. Demeure, "Ranger21: a synergistic deep learning optimizer," *arXiv preprint arXiv:2106.13731*, 2021.

[24] U. Asif, J. Tang, and S. Harrer, "Graspnet: An efficient convolutional neural network for real-time grasp detection for low-powered devices." in *IJCAI*, 2018, pp. 4875–4882.

[25] D. Guo, F. Sun, H. Liu, T. Kong, B. Fang, and N. Xi, "A hybrid deep architecture for robotic grasp detection," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1609–1614.

[26] F.-J. Chu, R. Xu, and P. A. Vela, "Real-world multiobject, multigrasp detection," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3355–3362, 2018.

[27] X. Zhou, X. Lan, H. Zhang, Z. Tian, Y. Zhang, and N. Zheng, "Fully convolutional grasp detection network with oriented anchor box," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7223–7230.

[28] H. Karaoguz and P. Jensfelt, "Object detection approach for robot grasp detection," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4953–4959.

[29] U. Asif, M. Bennamoun, and F. A. Sohel, "Rgb-d object recognition and grasp detection using hierarchical cascaded forests," *IEEE Transactions on Robotics*, 2017.

[30] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.