actuators

MDPI

Article

Fast UOIS: Unseen Object Instance Segmentation with Adaptive Clustering for Industrial Robotic Grasping

Kui Fu [1,2], Xuanju Dang [1,*], Qingyu Zhang [1] and Jiansheng Peng [2]

[1] School of Electronic and Automation, Guilin University of Electronic Technology, Guilin 541004, China
[2] School of Artificial Intelligence and Smart Manufacturing, Hechi University, Hechi 546300, China
* Correspondence: xjd69@guet.edu.cn

Abstract: Segmenting unseen object instances in unstructured environments is an important skill for robots to perform grasping-related tasks, where the trade-off between efficiency and accuracy is an urgent challenge to be solved. In this work, we propose a fast unseen object instance segmentation (Fast UOIS) method that utilizes predicted center offsets of objects to compute the positions of local maxima and minima, which are then used for selecting initial seed points required by the mean-shift clustering algorithm. This clustering algorithm that adaptively generates seed points can quickly and accurately obtain instance masks of unseen objects. Accordingly, Fast UOIS first generates pixel-wise predictions of object classes and center offsets from synthetic depth images. Then, these predictions are used by the clustering algorithm to calculate initial seed points and to find possible object instances. Finally, the depth information corresponding to the filtered instance masks is fed into the grasp generation network to generate grasp poses. Benchmark experiments show that our method can be well transferred to the real world and can quickly generate sharp and accurate instance masks. Furthermore, we demonstrate that our method is capable of segmenting instance masks of unseen objects for robotic grasping.

Keywords: adaptive clustering; unseen object instance segmentation; robotic grasping

check for updates

Citation: Fu, K.; Dang, X.; Zhang, Q.; Peng, J. Fast UOIS: Unseen Object Instance Segmentation with Adaptive Clustering for Industrial Robotic Grasping. *Actuators* **2024**, *13*, 305. https://doi.org/10.3390/act13080305

Academic Editor: Rocco Antonio Romeo

Received: 10 July 2024
Revised: 2 August 2024
Accepted: 7 August 2024
Published: 9 August 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Intelligent robot technology has always been the focus of research in the field of robotics, which can be applied to different real-world scenarios, such as healthcare, factories, and households [1,2]. In typical grasping tasks, the robot needs to perceive the environment through sensors and obtain the pose of the object to be grasped, as shown in Figure 1. However, designing efficient and accurate grasping algorithms in unstructured environments is an urgent problem to be solved due to the uncertainty of occlusion, object shape and pose. In particular, this requires that the robot can generalize previously learned knowledge to unseen objects. This paper focuses on unseen object instance segmentation to address the aforementioned issue, with the goal of quickly and accurately segmenting any arbitrary object instance in tabletop environments.

Traditional grasp algorithms mainly adopt analytical methods [3]. The methods use models based on geometry, kinematics, and dynamics to calculate feasible grasps. Due to the fact that these methods rely on modeling the physical interaction between robot end-effectors and objects, they often struggle to adapt well to unknown environments [4]. The data-driven approach is based on machine learning, which utilizes neural networks to learn feasible grasps from input data. Jiang et al. [5] proposed a two-step learning algorithm to learn a seven-dimensional oriented rectangle to represent the gripper configuration, including 3D position, 3D orientation, and gripper opening width. Thanks to the development of deep learning, object detection methods based on convolutional neural networks (CNNs) are used in grasp detection. Lenz et al. [6] proposed a two-step cascaded system based on deep learning, which utilizes two neural networks to detect potential grasps in a

coarse-to-fine manner. The system becomes very time-consuming due to the exhaustive search methods used to find potential grasp rectangles. Inspired by [7], subsequent work used anchor-based ideas to design two-stage [8,9] and single-stage [10,11] grasp detection models, respectively. In addition to using exhaustive search and anchor-based methods, other works [12,13] also used the grid division method to regress grasp rectangles.

Different from the direct regression-based methods for grasp rectangles, Morrison et al. [14] proposed a pixel-wise grasp generation model. The model can generate antipodal grasps from input images at real-time speed. Due to the lightweight design of the model architecture, there is still significant room for improvement in terms of accuracy. To make a trade-off between the performance and efficiency of the model, a lot of subsequent work [15–20] has been conducted. The above methods are mainly designed for single-object scenes, so they will be affected by non-target objects in cluttered scenes and they may generate infeasible grasps. Recent work adopted the multi-view idea by averaging multiple grasps [21] or choosing to generate grasps at the best view [22]. They achieved higher grasp success rates while also increasing computational costs. To further reduce the influence from non-target objects, other methods also used semantic segmentation [23] and instance segmentation [24] to improve the accuracy of grasp detection. For the grasping of unknown objects in cluttered scenes [25–28], recent work [25,27,28] proposed unseen object instance segmentation (UOIS) and applied it to robotic grasping.

This paper proposes a CNN-based tabletop object instance segmentation for object 4D pose generation, in which we try to solve the trade-off problem between speed and accuracy. Our scheme first generates semantic masks and 3D offsets to object centers from an organized point cloud, and then adopts a mean-shift algorithm to obtain instance masks. In order to obtain better initial seed points, previous work [25,28] used a distance-based algorithm to select them carefully. However, this method is time-consuming and does not yield stable results. To effectively solve this problem, we use an adaptive method to generate initial seed points. First, the positions of local maximum and minimum are calculated based on the predicted 3D offsets. These positions are then used to choose the initial seed points from the center votes. Compared with [25,28], the proposed method of adaptive initialization of seed points can greatly improve the efficiency of the entire segmentation pipeline and improve the overall performance to a certain extent. In particular, our method performs well in segmenting instance masks of tabletop objects, which is very beneficial for grasping tabletop objects. The main contributions of this paper are as follows:

- We propose a fast unseen object instance segmentation method and apply it to robotic grasping. Dual experiments on data sets and robots demonstrate the effectiveness of the proposed method.
- We propose a method for adaptive initialization of seed points to optimize the mean-shift clustering algorithm, which significantly improves the efficiency of the unseen object instance segmentation pipeline. Our method achieves competitive performance compared with benchmark methods.

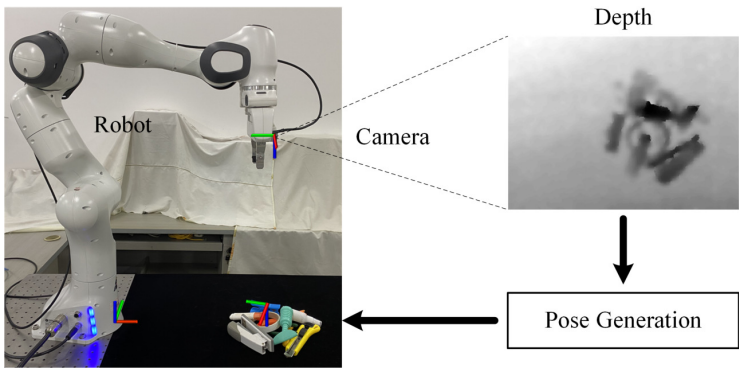


Figure 1. Typical robotic grasping system.

2. Problem Statement

A typical grasp configuration can be represented by a 5D grasp rectangle [6,8] in the image coordinates, which can be described as

$$\{x, y, \vartheta, w, h\} \tag{1}$$

where (x, y) represents the grasp center position, ϑ represents the grasp angle, w represents the width of the grasp rectangle, and h represents the height of the grasp rectangle.

Object detection-based grasp detectors usually use a 5D grasp representation, which makes it difficult for them to generate pixel-by-pixel grasps due to the time-consuming search of grasp rectangles. Instead of using the above 5D grasp representation, Morrison et al. [14] proposed an improved 5D grasp representation, which can be expressed in the robot frame as

$$g_r = \{\mathbf{p}_r, \theta_r, w_r, q\} \tag{2}$$

where $\mathbf{p}_r = (x, y, z)$ is the gripper's center position, θ_r is the gripper's rotation around the z-axis, w_r is the required gripper width, and q is the grasp quality score. The improved grasp representation can be described in the image space as

$$g_I = (\mathbf{p}_I, \theta_I, w_I, q) \tag{3}$$

where $\mathbf{p}_I = (u, v)$ represents the center point of the grasp in image coordinates, θ_I represents the rotation in the camera's reference frame, w_I represents the grasp width in image coordinates, and q represents the same scalar as in Equation (2).

q denotes the quality of the grasp at each point in the image. The value is a scalar between 0 and 1, where a value closer to 1 indicate higher grasp quality. θ_I denotes the angle of the grasp at each point. Because the grasp is symmetrical around $\pm\pi/2$ radians, the angle is expressed in $[-\pi/2, \pi/2]$. w_I denotes the required width at each point. The value is the range of [0, w_{max}] pixels where w_{max} represents the maximum width of the gripper, which can be converted to a physical measurement using the depth camera parameters and measured depth.

For a grasp obtained in the image space, it can be converted to the robot frame using the following transformations:

$$g_r = T_{RC}(T_{CI}(g_I)) \tag{4}$$

where T_{CI} converts the image space into 3D camera space using the intrinsic parameters of the camera, and T_{RC} converts 3D camera space into the robot frame using the calibration between the robot and the camera.

For a set of grasps in the image space, this can be expressed as

$$\mathbf{G}_I = \{\mathbf{Q}, \mathbf{\Theta}_I, \mathbf{W}_I\} \in \mathbb{R}^{3\times H\times W} \tag{5}$$

where $\mathbf{Q}$, $\mathbf{\Theta}_I$, and $\mathbf{W}_I$ represent three images composed of q, θ_I, and w_I at each pixel in an image, respectively.

The best grasp in the image space can be calculated as

$$g_I^* = \max_{\mathbf{Q}} \mathbf{G}_I \tag{6}$$

The best grasp g_r^* in the robot coordinates can be calculated by substituting Equation (6) into Equation (4). Typical planar grasp methods are susceptible to interference from nearby objects in cluttered object scenes. This may cause the robot to perform an incorrect grasp. We attempt to address the above problem using an unseen object instance segmentation model, which first segments possible instance masks from the input image. Then, the depth information that corresponds to the isolated masks or the closest-to-camera instance mask is extracted and fed into the grasp generator to generate pixel-wise grasps.

3. Method

Figure 2 shows the pipeline for segmenting unseen objects and generating grasp poses. The pipeline contains two main stages. In the first stage, Fast UOIS takes a three-channel organized point cloud $\mathcal{P}$ of XYZ coordinates as input and outputs instance masks of objects. It should be noted that $\mathcal{P}$ is calculated by backprojecting a depth image given camera intrinsics. In the second stage, the grasp generation algorithm takes the depth image and segmented instance masks as input and generates feasible grasps.

3.1. Network Architecture

In the first stage, instance masks of unseen objects are obtained from the input point cloud $\mathcal{P}$. We use an encoder–decoder architecture [25] based on U-Net [29], which performs instance segmentation by fusing features at different levels. Given an input point cloud $\mathcal{P}$ and features $\mathcal{F}_\eta(\mathcal{P})$ extracted from it, where $\mathcal{F}_\eta$ represents the encoder–decoder network with weights η. The semantic segmentation masks $\mathcal{S} \in \mathbb{R}^{H\times W}$ and the 3D offsets $\mathcal{V} \in \mathbb{R}^{H\times W\times 3}$ of the object centers can be calculated, respectively, as

$$\mathcal{H}_\phi : \mathcal{F}(\mathcal{P}) \rightarrow \mathcal{S} \tag{7}$$

$$\mathcal{M}_\gamma : \mathcal{F}(\mathcal{P}) \rightarrow \mathcal{V} \tag{8}$$

where $\mathcal{H}$ and $\mathcal{M}$ both represent a convolutional layer, their weights are ϕ and γ, respectively, and the semantic segmentation masks includes three classes: background, table, and foreground/object.

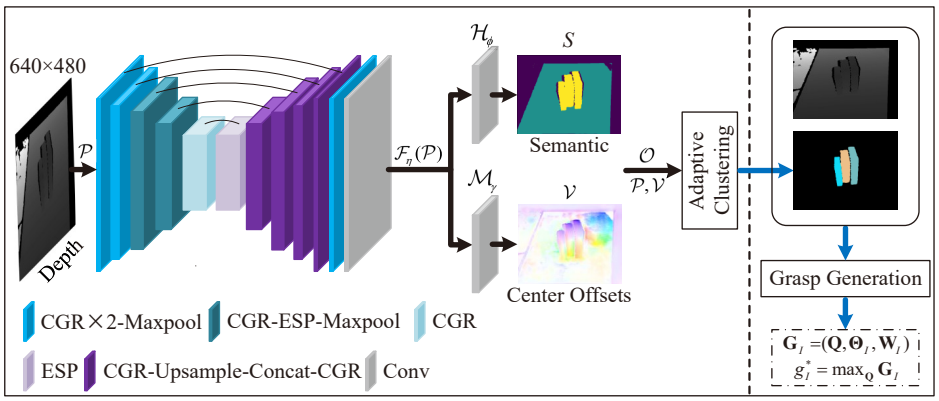


Figure 2. The pipeline for unseen object instance segmentation with adaptive clustering and grasp pose generation. CGR consists of a convolutional layer (Conv), a group normalization layer (GN) and a ReLU. ESP represents the efficient spatial pyramid modules [30].

Subsequently, the proposed adaptive clustering algorithm predicts pixel-wise classes from the input foreground/object masks $\mathcal{O}$, the point cloud $\mathcal{P}$, and the 3D offsets $\mathcal{V}$. The implementation details are described in Section 3.2.

3.2. Adaptive Clustering

To generate pixel-wise instance masks, we perform mean-shift clustering [31] on the predicted object centers $\mathcal{P} + \mathcal{V}$ in 3D space. Since $\mathcal{V}$ are 3D offsets, $\mathcal{P} + \mathcal{V}$ is the predicted object centers for each pixel. For simplicity, $\mathcal{P} + \mathcal{V}$ is called center votes $\mathcal{U}$. The mean-shift algorithm finds regions containing high-density data through a non-parametric method with a kernel density estimation. We use the kernel density estimation with a Gaussian kernel, which can be expressed as

$$\mathcal{Y}(\hat{x}) = \frac{1}{N}\sum_{n=1}^{N} K\left(\frac{||\hat{x} - \hat{x}_n||^2}{\sigma^2}\right) \tag{9}$$

where bandwidth $\sigma > 0$, Gaussian kernel $K(t) = \exp(-t^2/2\sigma^2)$, $\hat{x} \subset \mathbb{R}^D$ represents the seed points, and $\{\hat{x}_n\}_{n=1}^{N} \subset \mathbb{R}^D$ represents the data points to be clustered, which are the center votes $\mathcal{U}$.

For the seed points $\hat{x}$, their initial selection affect the performance and efficiency of the final clustering. In order to trade off the performance and efficiency of clustering, we design an algorithm for adaptively generating initial seed points. Considering a single object, the positions at the maximum and minimum Euclidean distances of its center offsets can be used to select initial seed points from the predicted centers. Due to the uncertainty of the number of objects in unstructured scenes, the seed points selected by the above method cannot be assigned to different objects. To further solve this problem, we generate initial seed points within a local window. Given the center votes $\mathcal{U}$, the initial seed points can be calculated as

$$\hat{x} = \mathcal{H}(\mathcal{U}, f(||\mathcal{V}_\mathcal{O}||_F) + g(||\mathcal{V}_\mathcal{O}||_F)) \tag{10}$$

where $\mathcal{V}_\mathcal{O} \in \mathbb{R}^{|\mathcal{O}|\times 2}$ represents the 2D center offsets of the foreground pixels, $f(\cdot)$ and $g(\cdot)$ denote the positions where the local maximum and minimum values of the given input are calculated, respectively, and $\mathcal{H}(r, c)$ represents taking out the points corresponding to positions of c in r.

Whether a certain position (u, v) is selectable can be determined by checking if its value in $||\mathcal{V}_\mathcal{O}||_F$ is equal to the local maximum. This process can be calculated as

$$P(u,v) = \begin{cases} \text{True, if } ||\mathcal{V}_\mathcal{O}||_F(u,v) = \mathrm{M}(||\mathcal{V}_\mathcal{O}||_F)(u,v) \\ \text{False, otherwise} \end{cases} \tag{11}$$

where $\mathrm{M}(\cdot)$ represents the max-pooling operation, whose kernel size and stride are $k \times k$ and 1, respectively. Equation (11) is used to select the position of the local maximum. When the result is true, it indicates that the position is retained for the initial seed point calculation using Equation (10). The selection of the position for the local minimum can be determined by computing $-||\mathcal{V}_\mathcal{O}||_F$ using Equation (11).

Figure 3 shows the process of adaptive initialization seed points. First, we calculate the Frobenius norm of the 2D center offsets $\mathcal{V}_\mathcal{O}$ shown in Figure 3a. Secondly, the max-pooling layer is used to calculate the local maximum and minimum values of $||\mathcal{V}_\mathcal{O}||_F$ and $-||\mathcal{V}_\mathcal{O}||_F$, respectively, and the seed points are determined through Equations (10) and (11). The results are shown in Figures 3b and 3c, respectively. Finally, the positions obtained above are combined, and the result is shown in Figure 3d. Compared with [25,28], the proposed initialization seed point algorithm can better allocate the points to different positions and adaptively set the number of initial seed points.

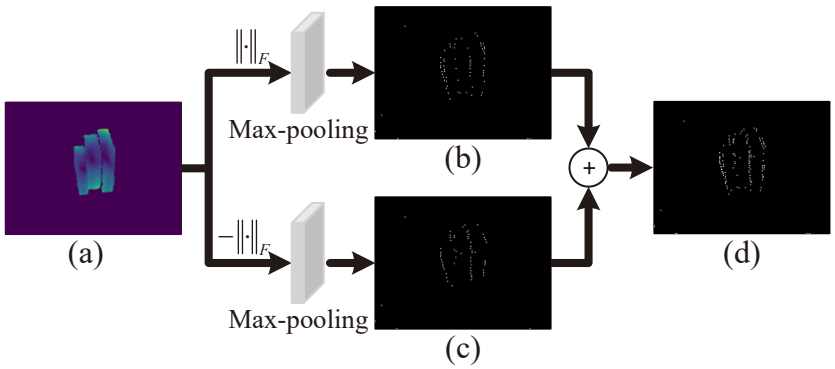


Figure 3. The method for adaptively generating seed points. (**a**) Center offsets of the foreground. (**b**) Positions of local maximum seed points. (**c**) Positions of local minimum seed points. (**d**) Positions of the initial seed points.

3.3. Loss Functions

Following [25], we use semantic segmentation loss $\mathcal{L}_{SS}$, center offset loss $\mathcal{L}_{CO}$, clustering loss $\mathcal{L}_{C}$, and separation loss $\mathcal{L}_{Sep}$ to train the model for generating semantic masks $\mathcal{S}$ and center offsets $\mathcal{V}$.

Semantic Segmentation Loss: We use weighted cross-entropy, which can be calculated as

$$\mathcal{L}_{SS} = \sum_i w_i \mathcal{L}_{CE}(\mathcal{S}_i, \hat{\mathcal{S}}_i) \tag{12}$$

where $\mathcal{S}_i$ and $\hat{\mathcal{S}}_i$ are the predicted and the ground truth probabilities of pixel i, respectively, and $\mathcal{L}_{CE}$ is the cross-entropy. The weight w_i is inversely proportional to the number of pixels with labels equal to $\hat{\mathcal{S}}_i$, normalized to sum to 1.

Center Offset Loss: We use a weighted smoothing L1 loss (Huber loss), which can be calculated as

$$\mathcal{L}_{CO} = \sum_i w_i \mathcal{L}_{H}(\mathcal{V}_i, \hat{\mathcal{V}}_i) \tag{13}$$

where $\mathcal{V}_i$ and $\hat{\mathcal{V}}_i$ denote the predicted and ground truth center offsets, respectively, the weight w_i is inversely proportional to the number of pixels with labels equal to $\hat{\mathcal{S}}_i$ in Equation (12), and $\mathcal{L}_H$ denotes the Huber loss.

Clustering Loss: We unfold the Gaussian mean-shift (GMS) for several iterations and perform it on the points to be clustered. GMS iteratively shifts a set of $D = |\mathcal{O}|$ 3D seed points $\hat{X} \in \mathbb{R}^{D\times 3}$ to higher-density regions. Let $\hat{X}^{(l)}$ denote the points at the lth iteration of GMS. $\hat{X}^{(0)}$ is initialized to the center votes $\mathcal{U}_{\mathcal{O}} \in \mathbb{R}^{|\mathcal{O}|\times 3}$ of the foreground pixels. We use the following loss to $\hat{X}^{(l)}$ and $\mathcal{U}_{\mathcal{O}}$, with corresponding object instance labels $Y \in \mathbb{R}^{|\mathcal{O}|}$:

$$\mathcal{L}_C^{(l)}(\hat{X}^{(l)}, \mathcal{Z}, Y) \quad = \sum_{i=1}^{D} \sum_{j\in\mathcal{O}} w_{ij} \mathbb{1}\{Y_i = Y_j\} d^2(\hat{X}_i^{(l)}, \mathcal{Z}_j) + w_{ij} \mathbb{1}\{Y_i \neq Y_j\} [\delta - d(\hat{X}_i^{(l)}, \mathcal{Z}_j)]_+^2 \tag{14}$$

where w_{ij} is inverse proportional weight with respect to class size, $d(\cdot, \cdot)$ is the Euclidean distance, and $[\cdot]_+ = max(\cdot, 0)$, $\mathbb{1}\{\cdot\}$ is the indicator function. For simplicity, $\mathcal{U}_{\mathcal{O}}$ is renamed to $\mathcal{Z}$. This loss function influences the kernel density estimation modes to be close to its point, and at least δ away from points not belonging to the cluster. δ is set to 0.1.

Applying Equation (14) to all points belonging to objects would result in excessive memory usage, so a stochastic version was used instead in [25,28]. They sampled a set $\mathcal{I} \subset \{1, 2, \cdots, |\mathcal{O}|\}$ containing $\mathcal{N}$ indices and set $\hat{X}^{(0)} = \mathcal{Z}_{\mathcal{I}}$ and ran GMS clustering only on these points. Different from [25,28], we use the method proposed in Section 3.2 to generate initial seed points. Similar to [25,28], when the number of generated initial seed points exceeds $\mathcal{N}$, we periodically sample a set containing $\mathcal{N}$ seed points. $\mathcal{N}$ is set to 50 and 300 during training and testing, respectively. We unroll GMS for L iterations and apply $\mathcal{L}_C^{(l)}$ at each iteration. L is set to 5 and 10 during training and testing, respectively.

Separation Loss: We employ a separation loss, which encourages the center votes to not necessarily be at the center of an object, as long as they are far from the center votes of other objects. To do this, consider the following equation:

$$M_{ij} = \frac{\exp(-\tau d(c_j, \mathcal{U}_i))}{\sum_{j'=1} \exp(-\tau d(c_{j'}, \mathcal{U}_i)} \tag{15}$$

where c_j is the jth ground truth object center, $i \in \mathcal{O}$, and $\tau > 0$ is a hyperparameter, which is set to 15. Equation (15) is simply the distance from center vote $\mathcal{U}_i$ to all object centers scaled by τ with a softmax applied. The separation loss uses cross-entropy, which can be expressed as

$$\mathcal{L}_{Sep}(M_{ij}) = -\sum_{j=1} \mathbb{1}\{Y_i = j\} \log(M_{ij}) \tag{16}$$

The total loss is denoted as $\mathcal{L}_{Total} = \lambda_{SS}\mathcal{L}_{SS} + \lambda_{CO}\mathcal{L}_{CO} + \lambda_C\mathcal{L}_C + \lambda_{Sep}\mathcal{L}_{Sep}$. The hyperparameter weights λ_{SS}, λ_{CO}, λ_C and λ_{Sep} are set to 3, 5, 1, and 1 respectively.

4. Experiment

4.1. Datasets

We trained our model using the tabletop object dataset (TOD) [25]. The TOD dataset is a non-photorealistic synthetic dataset consisting of 40k synthetic scenes of cluttered objects on a tabletop. When evaluating model performance, we used the OCID [32] and OSD [33] datasets constructed under real scenarios. OCID has 2346 RGB-D images with a size of 640 × 480, including cluttered tabletop and floor objects. To facilitate subsequent evaluation, we named the datasets in OCID that only contain cluttered objects on tabletop or floor as OCID-T and OCID-F, respectively. For the OCID-F dataset, the value of depth zero in each scene is set to 1600 mm to simulate a tabletop and named OCID-FT. OCID uses a semi-automatic approach to annotation, building incrementally by placing an additional object in each scene and using the depth difference between two consecutive images to mark new objects. The OSD has 110 RGB-D images with a size of 640 × 480. Objects are placed on a tabletop, and their annotations are manually labeled.

4.2. Metrics

To evaluate the performance of instance segmentation, we used precision (P), recall (R), and F-measure (F). Precision, recall, and F-measure are calculated as $P = \sum_i |b_i \cap g(b_i)| / \sum_i |b_i|$, $R = \sum_i |b_i \cap g(b_i)| / \sum_j |g_j|$ and $F = 2PR/(P+R)$, where b_i is the set of pixels belonging to predicted object i on the tabletop, $g(b_i)$ is the set of pixels of the matched ground truth object of b_i, and g_j is the set of pixels for ground truth object j. The above three metrics are called the overlap P/R/F, since the true positives are counted by the pixel overlap of the whole object. In addition, we also used the boundary P/R/F metric to evaluate how sharp the predicted boundary matches against the ground truth boundary. See [25] for more details. We reported all P/R/F measures in the range 0 to 100.

4.3. Details and Setups

The proposed model was implemented using 1.9.1 version of PyTorch. The model was trained using the Adam optimizer for 150k iterations, with a learning rate of 1×10^{-4} and a batch size of 8. During the training stage, we applied multiplicative gamma noise to enhance the depth images similar to [25,28], and add Gaussian process noise to the back-projected point clouds. The kernel size of the max-pooling layer for adaptive clustering was set to 13 × 13. The model was trained using a 2.4 GHz Intel Xeon Silver 4210R CPU and NVIDIA GeForce RTX3090 graphics card. During the testing stage, we removed any cluster of pixels that is smaller than 800. We set the kernel size of the max-pooling layer according to the object size of different datasets. For OCID and OSD, we set the kernel size of the max-pooling layer to 9 × 9 and 13 × 13, respectively. It should be noted that when using adaptive clustering, we removed the largest connected component in the post-processing stage, and other post-processing operations are consistent with [25,28]. In robotic grasping, we utilized a Franka Emika robotic arm with a Franka Hand. A RealSense D435i RGB-D camera was used to capture the required RGB-D images.

4.4. Experiments on Datasets

Quantitative Results: Table 1 compares the performance of different methods on OCID and OSD, where the bold numbers represent the best performances. "†" indicates that the results of this experiment are reported in [25]. "‡" indicates that the time is reported by the original paper, while the times required by other methods are obtained through our testing on the OSD dataset. UOIS-Net-3D-AC and FD-TNN-AC represent the experimental results obtained by applying our proposed adaptive clustering algorithm to the training models provided by [25,28], respectively. It should be clarified that when applying UOIS-Net-3D-AC to test the OCID dataset, we did not simulate tabletop scenes. Ours-OC represents applying the initialization seed points method used in [25] to the model trained by our proposed method. Ours-AC represents that the experimental results are obtained after training and testing our proposed adaptive clustering algorithm. UOIS-Net-3D-AC has a

slight decrease in performance compared to the original methods, but the computational consumption is roughly reduced by half. Specifically, UOIS-Net-3D-AC achieved a 0.7% improvement in the overlap F-measure and a 0.6% improvement in the boundary F-measure. In addition, Ours-OC and Ours-AC achieved better scores than UOIS-Net-3D in various F-measures on OCID and OSD, which demonstrated the effectiveness of the adaptive clustering algorithm. In particular, FD-TNN-AC with an adaptive clustering approach achieved state-of-the-art performance. Compared with the state-of-the-art FD-TNN, FD-TNN-AC can achieve better performance with nearly half the computational cost. On OCID, FD-TNN-AC achieved relative improvements of 0.8% and 1.2% in overlap F-measure and boundary F-measure compared to FD-TNN, respectively. On OSD, FD-TNN-AC performed better than FD-TNN in overall performance.

Table 1. Evaluation of our method against SOTA methods on OCID and OSD.

Method	Time (s)	Parm (M)	OCID						OSD					
			Overlap			Boundary			Overlap			Boundary		
			P	R	F	P	R	F	P	R	F	P	R	F
Mask R-CNN † [34]	-	-	82.7	78.9	79.9	79.4	67.7	71.9	73.8	72.9	72.2	49.6	40.3	43.1
UOIS-Net-3D [25]	0.120	22.3	88.8	89.2	88.8	86.9	80.9	83.5	85.7	82.1	83.0	74.3	67.5	70.0
UCN [26]	0.250 ‡	-	83.1	90.7	86.4	77.7	74.3	75.6	78.7	83.8	81.0	52.6	50.0	50.9
UOAIS-Net [35]	0.074	77.4	89.9	90.9	89.8	86.7	**84.1**	84.7	84.9	86.4	85.5	68.2	66.2	66.9
MSMFormer [27]	0.278	-	88.4	90.2	88.5	84.7	83.1	83.0	79.5	86.4	82.8	53.5	71.0	60.6
FD-TNN [28]	0.123	27.2	90.0	89.9	89.7	86.4	83.1	84.4	88.0	**87.4**	87.7	74.3	72.8	73.4
UOIS-Net-3D-AC	0.065	22.3	88.8	88.7	88.5	86.7	80.5	83.1	86.3	82.6	83.7	74.7	68.1	70.6
FD-TNN-AC	0.068	27.2	**90.4**	**91.0**	**90.5**	**88.1**	83.9	**85.6**	**88.2**	87.2	**87.7**	**75.0**	**72.9**	**73.7**
Ours-OC	0.120	22.3	89.0	89.4	89.0	86.9	81.2	83.7	87.0	84.9	85.9	73.2	69.1	70.8
Ours-AC	0.065	22.3	89.1	89.4	89.0	86.8	81.2	83.6	87.5	85.3	86.2	73.8	69.7	71.4

Figure 4 shows the stability of different methods on OCID. For FD-TNN and UOIS-Net-3D, we tested the results 10 times each and took their average as the final result. It can be seen from Figure 4 that the results of FD-TNN and UOIS-Net-3D fluctuate to a certain extent due to the selection of the initial seed points of the clustering algorithm, and the maximum absolute error is about 0.2%. FD-TNN-AC and Ours-AC achieved stable results every time and outperformed the corresponding original methods overall.

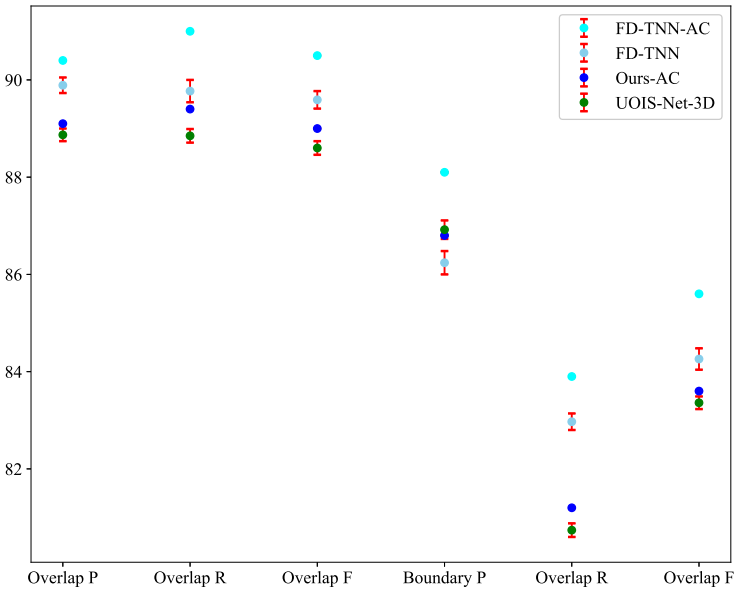
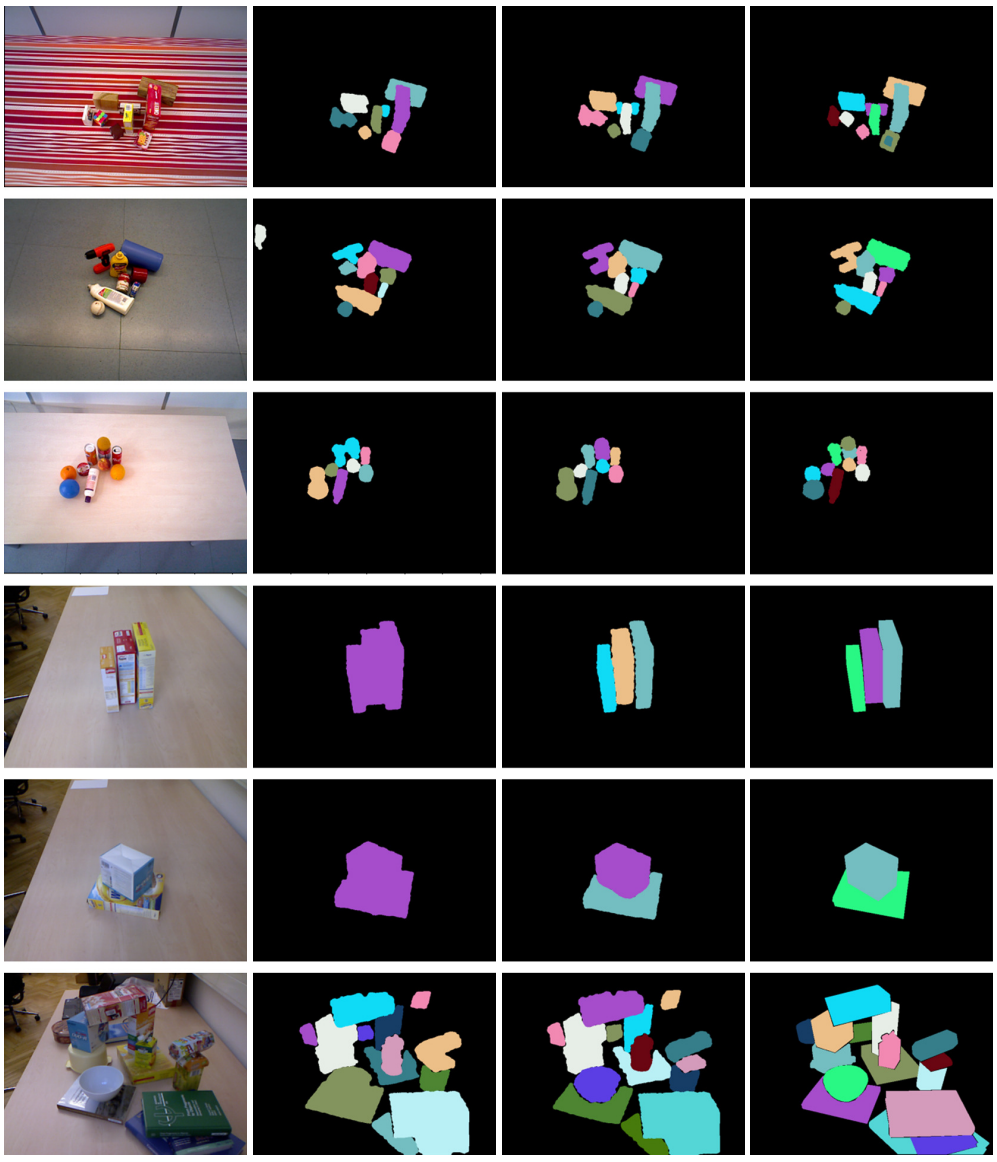


Figure 4. Stability of different methods.

To evaluate the performance of the proposed method for segmentation instance masks of objects on tabletops, we conducted instance segmentation tests on the OCID-F and OCID-FT datasets respectively, with results shown in Table 2. The experimental results in Table 2 show that the performance of FD-TNN-AC is better for tabletop objects than for floor objects. Compared with the results obtained on OCID-F, the overlap F-measure and boundary F-measure obtained on OCID-FT are relatively improved by 2.5% and 3.8%, respectively. This performance is also reflected in the results that Ours-AC achieved on the OCID-F and OCID-FT datasets. Overall, although our method was affected by partial data from floor scenes, this issue could be mitigated to some extent by simulating floor scenes as tabletop scenes.

Qualitative Results: Figure 5 shows the results of segmentation instance masks on OCID and OSD by Ours-AC and UOIS-Net-3D. Rows 1 to 3 of Figure 5 show the results on OCID. The other rows show the results on the OSD. The results of Figure 5b,c are from UOIS-Net-3D and Ours-AC, respectively. Our method demonstrates the ability to accurately segment instance masks when objects are close to each other on both OCID and OSD datasets.



(a) (b) (c) (d)

Figure 5. Qualitative results on OCID and OSD. (**a**) RGB. (**b**) Results of UOIS-Net-3D. (**c**) Results of Ours-AC. (**d**) Ground truths.

Table 2. Evaluation of our method on OCID-F and OCID-FT.

Method	Dataset	OCID					
		Overlap			Boundary		
		P	R	F	P	R	F
FD-TNN-AC	OCID-F	88.0	87.1	87.0	82.5	79.7	80.6
FD-TNN-AC	OCID-FT	89.5	90.2	89.5	87.4	82.2	84.4
Ours-AC	OCID-F	88.0	84.5	85.6	82.5	77.8	79.4
Ours-AC	OCID-FT	87.6	89.1	88.1	85.5	79.6	82.0

Impact of k-Value: In Table 3, we test the sensitivity of Ours-AC to the settings of kernel size of k over $\{9, 11, 13, 15, 17\}$.The value of k determines the number of initial seed points. When k is too small, the number of generated initial seed points will increase, which is beneficial to improving model performance to a certain extent, i.e., F-measure scores on OCID. However, this also results in increased computational costs. On the other hand, when k is too large, the number of generated seeds will decrease. Although this will reduce computational costs, it is not conducive to improving model performance. In fact, the value of k can be set relatively reasonably based on the size of objects in the image.

Table 3. Evaluation of different kernel sizes.

$k \times k$	OCID						OSD					
	Overlap			Boundary			Overlap			Boundary		
	P	R	F	P	R	F	P	R	F	P	R	F
9×9	89.06	89.44	89.02	86.85	81.21	83.60	87.53	85.36	86.22	73.88	69.74	71.46
11×11	89.03	89.42	89.00	86.79	81.20	83.57	87.52	85.37	86.21	73.88	69.79	71.49
13×13	89.00	89.40	88.98	86.74	81.20	83.55	87.52	85.37	86.21	73.88	69.79	71.49
15×15	88.93	89.33	88.91	86.63	81.15	83.47	87.47	85.33	86.17	73.67	69.65	71.31
17×17	88.90	89.30	88.87	86.58	81.12	83.43	87.46	85.33	86.17	73.69	69.67	71.33

4.5. Robotic Grasping

We applied Ours-AC to the Franka Emika robotic arm with a Franka hand and a wrist-mounted RGB-D camera for grasping unseen objects in cluttered scenes (Figure 6). It should be clarified that we remove any clusters smaller than 500 pixels in the post-processing stage to accommodate grasp scenes. The task of the robot was to grasp objects on the table and then place them in a bin. The proposed method was first used to segment object instances from camera-captured images, and then the depth information corresponding to the isolated instance masks was fed into the grasp generation model. When the isolated instance masks were not available, the depth information corresponding to the instance mask closest to the camera was fed into the grasp generation model. The grasp model used GG-CNN [14] trained on the Cornell dataset. The robot setup and grasping process are kept consistent with existing work [14,15,28] for a fair comparison. Before each grasp attempt, the camera was positioned approximately 520 mm above and parallel to the surface of the table to obtain the depth image of the objects to be grasped. To perform a grasp, the robot first moved to a pre-grasping position. Subsequently, the robot moved straight down until the grasp pose was met or a collision was detected via force feedback in the robot. Finally, the gripper was closed, lifted, and moved to the placement area to place the object.

The 10 randomly selected objects in Figure 6a were shaken in a box and emptied into the robot's workspace for each grasp attempt. The robot attempted multiple grasps, and any objects that were grasped were removed. This process would continue until all objects were grasped, three consecutive object grasps were failures or all objects were outside the observable workspace. The results are shown in Table 4, where the bold numbers represent the best performances. The results of other algorithms are reported in their original papers. "†" represents the time required for this method, which was obtained from our testing. We

ran this experiment 10 times, and the robot achieved a grasp success rate of 93.4% (99/106) compared with 87%(83/96) in Morrison et al. [14]. Compared to the current state-of-the-art method [28], our grasp success rate is 3.6% lower. However, our grasp method reduces the time consumption by 60 ms. It should be noted that using only depth maps makes our grasp model less dependent on backgrounds to a large extent. In addition, using instance segmentation to weaken the complexity of the grasping environment enabled our method to handle more complex and cluttered environments, thus avoiding the limitation of grasp methods to simple scenarios with multiple objects to some extent [15].



(a) (b) (c)

Figure 6. Objects for robotic grasping. (**a**) The objects were used to reproduce the grasping in the clutter experiment by Morrison et al. [14]. (**b**) 17 household objects. (**c**) 17 adversarial objects from Mahler et al. [36].

Figure 7 shows some instance segmentation results and snapshots of the grasping states. A demonstration video of robotic experiments can be found here https://reurl.cc/E4znAk (accessed on 23 December 2023). Our method was able to accurately segment instance masks of objects in most cases. However, there were also instances of segmentation failures, such as under-segmentation of objects that were too close to each other. In addition, we created an object set comprising 27 household objects (Figure 6a,b) and 17 adversarial objects (Figure 6c). For each experimental run, we randomly selected 10 adversarial and 10 household objects to mix and emptied them onto the table in front of the robot. We ran this experiment 10 times, and the robot achieved a grasp success rate of 82.4% (183/222) compared with the 68% (134/196) in Morrison et al. [14]. The experimental results demonstrate that our method is robust in real-world grasping experiments and has the potential for related robotic grasping applications.

Table 4. Grasp success rate in cluttered scenes.

Method	Gripper	GPU	Time (ms)	Grasp Success Rate (%)
2020/Morrison et al. [14]	Parallel-jaw	RTX3090	12 †	87
2020/Kumra et al. [15]	Parallel-jaw	RTX3090	14 †	93.5
2022/Wu et al. [37]	Parallel-jaw	RTX2080	26	83.3
2022/Wang et al. [38]	Parallel-jaw	RTX3090	142 †	93.5
2022/Liu et al. [9]	Parallel-jaw	RTX2080Ti	47	90.2
2023/Cao et al. [19]	Parallel-jaw	RTX2080Ti	-	85.9
2023/Yu et al. [8]	Parallel-jaw	TITAN RTX	40	91.7
2024/Fu et al. [28]	Parallel-jaw	RTX3090	137	**97.0**
Ours	Parallel-jaw	RTX3090	77	93.4

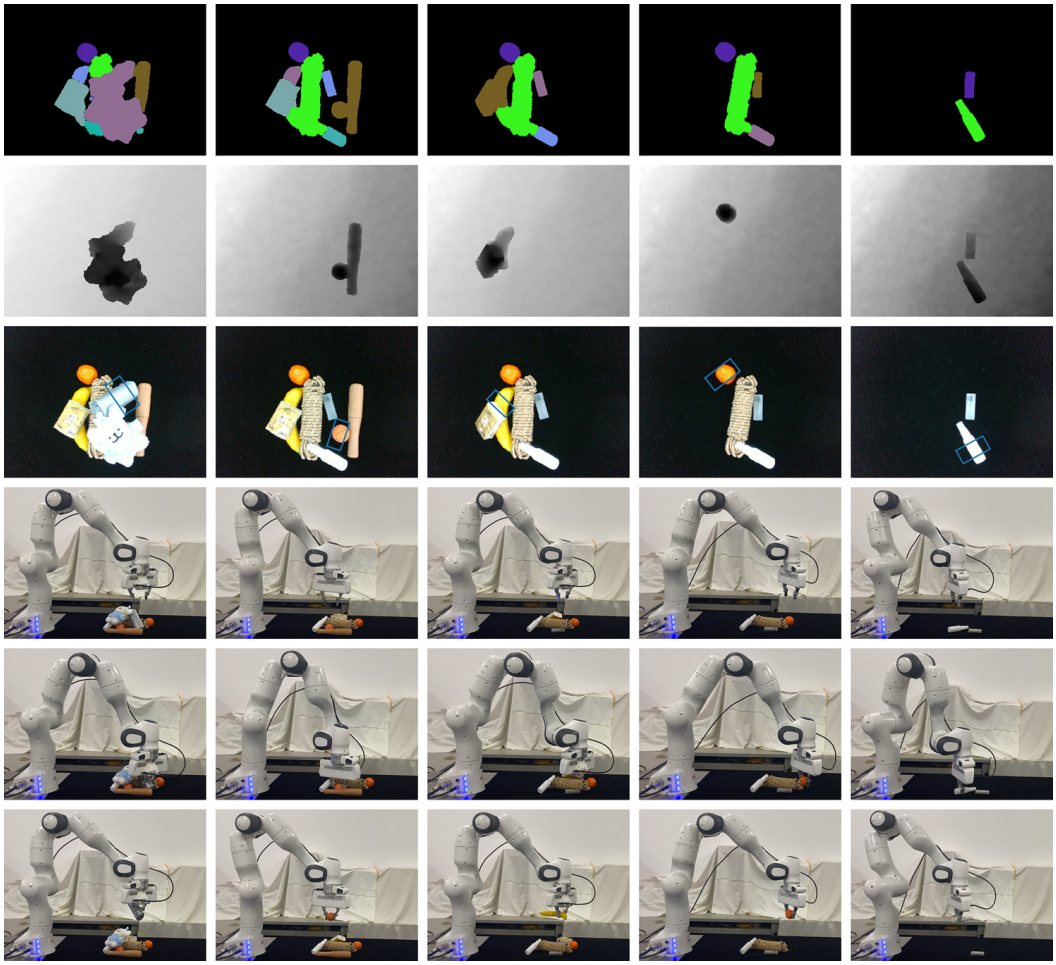


Figure 7. Visualization of emptying cluttered objects using instance segmentation. Rows 1 to 3 represent the segmented instance masks, the depth extracted from the masks, and the generated grasp rectangle, respectively. Rows 4 to 6 represent the states of pre-grasping, grasping and lifting, respectively.

4.6. Discussion of Failure Cases

In this section, further analysis is conducted on experimental results of segmenting unseen object instances. Compared with benchmark methods, the proposed approach was able to generate more accurate segmentation results to some extent when objects were close together or overlapping. However, there were also several typical cases of segmentation failures. The main failures involved under-segmentation, such as in the results of the third and sixth rows in Figure 5 and the first row in Figure 7. In these examples, the proposed method failed to segment two similar spherical objects or two overlapping books. Another scenario was over-segmentation, as seen in the result of the sixth row in Figure 5. Future work could explore efficient methods using multi-view images or combining multimodal information (e.g., RGB images) to address these challenges. In practical scenarios, research could focus on robot manipulation strategies to further address these issues, e.g., push–grasp synergy methods.

5. Conclusions

This paper proposes a fast segmentation of unseen object instances and demonstrates grasping on a real robot. The proposed method uses an adaptive clustering algorithm by calculating the positions of local maximum and minimum from the predicted center offsets and obtaining the points at the corresponding positions in the center votes as the initial seed points. The experimental results demonstrate that the proposed method can achieve comparable performance to state-of-the-art methods with less computational costs. In particular, we show the superiority of our method for segmenting instance masks of

unseen tabletop objects. The proposed method also demonstrates the effectiveness of grasping unknown objects in cluttered scenes.

In the future, we plan to optimize the algorithm for deployment on embedded AI computing devices. Additionally, we will further optimize the model to improve the accuracy and speed of segmenting unseen object instances. Furthermore, 4D pose representations generally have faster inference speeds than 6D pose representations, but this also makes the estimated grasp poses more restricted in space. Therefore, we will develop a 6D pose estimation algorithm in the next phase to enable more diverse grasping in the world space.

Author Contributions: Conceptualization, K.F.; methodology, K.F.; software, K.F. and Q.Z.; validation, K.F., X.D. and Q.Z.; formal analysis, K.F.; investigation, K.F., X.D. and Q.Z.; resources, K.F. and Q.Z.; writing—original draft preparation, K.F., Q.Z. and J.P.; writing—review and editing, K.F. and J.P.; visualization, K.F. and Q.Z.; supervision, X.D.; project administration, X.D.; funding acquisition, X.D. and K.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 62263004, in part by Science and Technology Plan Project of Guangxi under Grant AA24010001, and in part by Research Project of Hechi University under Grant 2023XJYB009.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors gratefully appreciate the equipment provided by the Guangxi Key Laboratory of Sericulture Ecology and Applied Intelligent Technology, and the Guangxi Colleges and Universities Key Laboratory of AI and Information Processing.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Levine, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; Quillen, D. Learning hand-eye coordination for robotic grasping with large-scale data collection. *Int. J. Exp. Robot. Res.* **2018**, *37*, 421–436. [CrossRef]
2. Zeng, A.; Song, S.; Yu, K.T.; Donlon, E.; Hogan, F.R.; Bauza, M.; Ma, D.; Taylor, O.; Liu, M.; Romo, E.; et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *Int. J. Exp. Robot. Res.* **2022**, *41*, 690–705. [CrossRef]
3. Bicchi, A.; Vijay, K. Robotic grasping and contact: A review. In Proceedings of the IEEE International Conference on Robotic Automation, San Francisco, CA, USA, 24–28 April 2000; pp. 348–353.
4. Rubert, C.; Kappler, D.; Morales, A.; Schaal, S.; Bohg, J. On the relevance of grasp metrics for predicting grasp success. In Proceedings of the IEEE International Conference on Robotic Automation, Vancouver, BC, Canada, 24–28 September 2017; pp. 265–272.
5. Jiang, Y.; Moseson, S.; Saxena, A. Efficient grasping from rgbd images: Learning using a new rectangle representation. In Proceedings of the IEEE International Conference on Robotic Automation, Shanghai, China, 9–13 May 2011; pp. 3304–3311.
6. Lenz, I.; Lee, H.; Saxena, A. Deep learning for detecting robotic grasps. *Int. J. Robot. Res.* **2015**, *34*, 705–724. [CrossRef]
7. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, **2016**, *39*, 1137–1149. [CrossRef] [PubMed]
8. Yu, S.; Zhai, D.-H.; Xia, Y. CGNet: Robotic grasp detection in heavily cluttered scenes. *IEEE/ASME Trans. Mech.* **2023**, *28*, 884–894. [CrossRef]
9. Liu, D.; Tao, X.; Yuan, L.; Du, Y.; Cong, M. Robotic objects detection and grasping in clutter based on cascaded deep convolutional Neural Network. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–10. [CrossRef]
10. Zhou, X.; Lan, X.; Zhang, H.; Tian, Z.; Zhang, Y.; Zheng, N. Fully convolutional grasp detection network with oriented anchor box. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, Spain, 1–5 October 2018; pp. 7223–7230.
11. Song, Y.; Gao, L.; Li, X.; Shen, W. A novel robotic grasp detection method based on region proposal networks. *Robot. Comput.-Integr. Manuf.* **2020**, *65*, 101963 [CrossRef]
12. Redmon, J.; Angelova, A. Real-time grasp detection using convolutional neural networks. In Proceedings of the IEEE International Conference on Robotic Automation, Seattle, WA, USA, 26–30 May 2015; pp. 1316–1322.
13. Yu, Q.; Shang, W.; Zhao, Z.; Cong, S.; Li, Z. Robotic grasping of unknown objects using novel multilevel convolutional neural networks: From parallel gripper to dexterous hand. *IEEE Trans. Automat. Sci. Eng.* **2021**, *18*, 1730–1741. [CrossRef]
14. Morrison, D.; Corke, P.; Leitner, J. Learning robust, real-time, reactive robotic grasping. *Int. J. Robot. Res.* **2020**, *39*, 183–201. [CrossRef]

15. Kumra, S.; Joshi, S.; Sahin, F. Antipodal robotic grasping using generative residual convolutional neural network. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 9626–9633.
16. Niu, M.; Lu, Z.; Chen, L.; Yang, J.; Yang, C. VERGNet: Visual enhancement guided robotic grasp detection under low-light condition. *IEEE Robot. Automat. Lett.* **2023**, *8*, 8541–8548. [CrossRef]
17. Tian, H.; Song, K.; Li, S.; Ma, S.; Yan, Y. Lightweight pixel-wise generative robot grasping detection based on RGB-D dense fusion. *IEEE Trans. Instrum. Meas.* **2022**, *7*, 5017912. [CrossRef]
18. Yu, S.; Zhai, D.-H.; Xia, Y.; Wu, H.; Liao, J. SE-ResUNet: A novel robotic grasp detection method. *IEEE Robot. Automat. Lett.* **2022**, *7*, 5238–5245. [CrossRef]
19. Cao, H.; Chen, G.; Li, Z.; Feng, Q.; Lin, J.; Knoll, A. Efficient grasp detection network with gaussian-based grasp representation for robotic manipulation. *IEEE/ASME Trans. Mech.* **2023**, *28*, 1384–1394. [CrossRef]
20. Fu, K.; Dang, X. Light-weight convolutional neural networks for generative robotic grasping. *IEEE Trans. Ind. Inform.* **2024**, *20*, 6696–6707. [CrossRef]
21. Wu, Y.; Fu, Y.; Wang, S. Information-theoretic exploration for adaptive robotic grasping in clutter based on real-time pixel-level grasp detection. *IEEE Trans. Ind. Electron.* **2024**, *71*, 2683–2693. [CrossRef]
22. Kasaei, H.; Kasaei, M. MVGrasp: Real-time multi-view 3D object grasping in highly cluttered environments. *Robot. Auton. Syst.* **2023**, *160*, 104313. [CrossRef]
23. Ainetter, S.; Fraundorfer, F. End-to-end trainable deep neural network for robotic grasp detection and semantic segmentation from rgb. In Proceedings of the IEEE International Conference on Robotic Automation, Xi'an, China, 30 May 2021–5 June 2021; pp. 13452–13458.
24. Yan, Y.; Tong, L.; Song, K.; Tian, H.; Man, Y.; Yang, W. SISG-Net: Simultaneous instance segmentation and grasp detection for robot grasp in clutter. *Adv. Eng. Inform.* **2023**, *58*, 102189. [CrossRef]
25. Xie, C.; Xiang, Y.; Mousavian, A.; Fox, D. Unseen object instance segmentation for robotic environments. *IEEE Trans. Robot.* **2021**, *37*, 1343–1359. [CrossRef]
26. Xiang, Y.; Xie, C.; Mousavian, A.; Fox, D. Learning rgb-d feature embeddings for unseen object instance segmentation. In Proceedings of the Conference on Robot Learning, Cambridge, MA, USA, 16–18 November 2020; pp. 461–470.
27. Lu, Y.; Chen, Y.; Ruozzi, N.; Xiang, Y. Mean Shift Mask Transformer for Unseen Object Instance Segmentation. *arXiv* **2022**, arXiv:2211.11679.
28. Fu, K.; Dang, X.; Zhang, Y. Taylor neural network for unseen object instance segmentation in hierarchical grasping. *IEEE/ASME Trans. Mech.* **2024**. [CrossRef]
29. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, 5–9 October 2015; pp. 234–241.
30. Mehta, S.; Rastegari, M.; Caspi, A.; Shapiro, L.; Hajishirzi, H. ESPnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 552–568.
31. Carreira-Perpinan, M.A. A review of mean-shift algorithms for clustering. *arXiv* **2015**, arXiv:1503.00687.
32. Suchi, M.; Patten, T.; Vincze, M. EasyLabel: A semi-automatic pixelwise object annotation tool for creating robotic rgb-d datasets. In Proceedings of the IEEE International Conference on Robotic Automation, Montreal, QC, Canada, 20–24 May 2019; pp. 6678–6684.
33. Richtsfeld, A.; Mörwald, T.; Prankl, J.; Zillich, M.; Vincze, M. Segmentation of unknown objects in indoor environments. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 4791–4796.
34. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R.; Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
35. Back, S.; Lee, J.; Kim, T.; Noh, S.; Kang, R.; Bak, S.; Lee, K. Unseen object amodal instance segmentation via hierarchical occlusion modeling. In Proceedings of the IEEE International Conference on Robotic Automation, Philadelphia, PA, USA, 23–27 May 2022; pp. 5085–5092.
36. Mahler, J.; Liang, J.; Niyaz, S.; Laskey, M.; Doan, R.; Liu, X.; Ojea, J.A.; Goldberg, K. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv* **2017**, arXiv:1703.09312.
37. Wu, Y.; Zhang, F.; Fu, Y. Real-time robotic multigrasp detection using anchor-free fully convolutional grasp detector. *IEEE Trans. Ind. Electron.* **2022**, *69*, 13171–13181. [CrossRef]
38. Wang, D.; Liu, C.; Chang, F.; Li, N.; Li, G. High-performance pixel-level grasp detection based on adaptive grasping and grasp-aware network, *IEEE Trans. Ind. Electron.* **2022**, *69*, 11611–11621. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.