




Article

Depth Image–Based Deep Learning of Grasp Planning for Textureless Planar-Faced Objects in Vision-Guided Robotic Bin-Picking

Ping Jiang ^{*,†}, Yoshiyuki Ishihara [†], Nobukatsu Sugiyama [†], Junji Oaki, Seiji Tokura and Atsushi Sugahara and Akihito Ogawa

Corporate Research & Development Center, Toshiba Corporation, 1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan; yoshiyuki.ishihara@toshiba.co.jp (Y.I.); nobukatsu.sugiyama@toshiba.co.jp (N.S.); junji.ooaki@toshiba.co.jp (J.O.); seiji.tokura@toshiba.co.jp (S.T.); atsushi.sugahara@toshiba.co.jp (A.S.); akihito.ogawa@toshiba.co.jp (A.O.)

* Correspondence: ping2.jiang@toshiba.co.jp

† These authors contributed equally to this work.

Received: 20 December 2019; Accepted: 23 January 2020; Published: 28 January 2020



Abstract: Bin-picking of small parcels and other textureless planar-faced objects is a common task at warehouses. A general color image–based vision-guided robot picking system requires feature extraction and goal image preparation of various objects. However, feature extraction for goal image matching is difficult for textureless objects. Further, prior preparation of huge numbers of goal images is impractical at a warehouse. In this paper, we propose a novel depth image–based vision-guided robot bin-picking system for textureless planar-faced objects. Our method uses a deep convolutional neural network (DCNN) model that is trained on 15,000 annotated depth images synthetically generated in a physics simulator to directly predict grasp points without object segmentation. Unlike previous studies that predicted grasp points for a robot suction hand with only one vacuum cup, our DCNN also predicts optimal grasp patterns for a hand with two vacuum cups (left cup on, right cup on, or both cups on). Further, we propose a surface feature descriptor to extract surface features (center position and normal) and refine the predicted grasp point position, removing the need for texture features for vision-guided robot control and sim-to-real modification for DCNN model training. Experimental results demonstrate the efficiency of our system, namely that a robot with 7 degrees of freedom can pick randomly posed textureless boxes in a cluttered environment with a 97.5% success rate at speeds exceeding 1000 pieces per hour.

Keywords: deep learning; bin picking; grasp planning; textureless; visual servoing

1. Introduction

Demand for logistics workers in Japan is high and still rising, making automatic robot manipulation systems crucially important for overcoming labor shortages [1]. Picking textureless planar-faced objects is a common robot manipulation task for item transfer in warehouses.

Vision-guided robot picking tasks entail grasp planning to determine a target object and detect its grasp points, which are then used to track and pick the target object by visual servoing (VS) [2], an image-based visual-guided robot control scheme. Examples of VS include position-based VS [3,4] and hybrid VS [5–7]), which is flexible due to its robustness to camera calibration errors. VS generally requires a textured goal image for feature matching to estimate the relative pose of an object with respect to the camera [8], but many items in a warehouse, such as cardboard boxes, are textureless. Even for textured items, it is impractical and time-consuming to prepare textured goal images of various items in a warehouse. Bin picking in a warehouse is even more challenging, because items

in a bin are often in disorganized heaps, making it difficult to infer their relative poses in textured color images due to sensor noise, obstructions, and occlusions [9]. Therefore, using depth images, thereby eliminating the need for color images and extraction of image texture features, is preferred for picking planar-faced objects in cluttered environments.

In combination with VS, grasp planning determines the grasp order and grasp points of a target object. The graspability scores of grasp points are ranked, and an optimal target is selected as the target point for the grasp attempt. Grasp planning methods can be broadly divided into one- and two-stage methods. Two-stage methods first conduct object segmentation and pose estimation, then search for grasp points on the target object. Object segmentation can be conducted by traditional template matching based on designed descriptors (e.g., scale-invariant feature transform (SIFT) [10,11]) or deep convolutional neural networks (e.g., mask region-based CNN (R-CNN) [12], Single shot multibox detector (SSD) [13], You Only Look Once (YOLO) [14], or SegNet [15] for 2D object segmentation, and PointNet [16] or PointCNN [17] for 3D object segmentation). Given the segmented points, object pose can be estimated by cloud point registration, which searches a spatial transform to align two cloud sets [18]. Drost et al. [19] proposed Point Pair Features (PPF) for building a hash table as object model global descriptors and recovering 6D poses from a depth image by a voting scheme, the performance of which under noise and occlusion was improved in References [20,21], and Reference [18]. Recently, deep convolutional neural networks that jointly output the semantic segmentation and object 6D pose have been proposed [18,22–24]. These methods integrate object segmentation and pose estimation, and outperform conventional methods in cluttered scenes. Once the object pose is obtained and the object is known, a template-based method can be used to find corresponding grasp points predefined in a database, but such databases are not always affordable in a warehouse. As an alternative, analytical methods can analyze kinematics and dynamics of grasp candidates to rank the best grasp points [25], but doing so requires foreknowledge of certain criteria, such as object geometry, physics models, and force analytics [26,27], which might be unavailable or difficult to design for bin picking in cluttered scenes.

Unlike two-stage methods, a one-stage method [28–37], namely, one-shot grasp detection, directly regresses grasp points and their classes without object segmentation or pose estimation. This method is preferable for object picking in a warehouse for two reasons. First, in most cases of random bin picking, the goal is to grasp and place as many objects as possible per hour rather than to grasp a specific object in a pile, so segmentation is unnecessary. Second, one-stage methods are faster than two-stage methods. One-shot detection methods are thus best suited to real-time grasp detection [27]. Levine et al. [29] were among the first to incorporate a convolutional neural network (CNN) model that directly predicts the probability of success of a given motor command in visually guided grasp planning. Douglas et al. [36] proposed Generative Grasping Convolutional Neural Networks (GG-CNN) to predict the quality and pose of grasps at every pixel with fast prediction speeds of 19 ms. Zeng et al. [37] directly predicted the affordance map of four primitive grasp actions based on RGB-D images. However, these studies required large datasets. Johns et al. [5] showed that the major challenge with deep learning is the need for very large amounts of training data, and thus they opted to generate and use simulated data for the training process [38].

Faced with this challenge, recent works [30–35] have trained CNN models by data generated in simulations. Josh et al. [30] used domain randomization and generative models to predict scores for grasp candidates, but only tested their method on single-object grasping. Bousmalis et al. [31] proposed GraspGAN, a pixel-level domain adaptation model, to predict the grasp probability of a motion command input for bin picking, but this model's success rate is still relatively low. Mahler et al. [32,33] proposed Grasp-Quality CNN (GQ-CNN) to predict grasp robustness and to rank most grasp points. They generated grasp points in a simulator and evaluated the robustness of each using a metric named Robust Ferrari-Canny. Alexandre et al. [34] extended this method to a Grasp Quality Spatial Transformer Network (GQ-STN), which improves the precision of grasp robustness predictions. Ulrich et al. [35] proposed a CNN model that uses simulated depth images to predict the remaining

cost-to-go of moving to the nearest viable grasp, but execution times were so long that each grasp attempt took 20–30 s.

Although the above studies have proposed CNN models that are trained by simulated images to predict grasp points, the predictions are affected by differences between simulated and actual images. Domain randomization is effective but cannot totally eliminate the influences of unrealistic datasets. Further, these studies used a two-finger gripper or a one-vacuum-cup suction hand. To the best of our knowledge, no previous studies have used CNN models to predict grasp patterns (vacuum cup activation selection) for a two-vacuum-cup suction hand.

In this paper, we propose a novel depth-image-based vision-guided robot bin-picking system for textureless planar-faced objects for robots with a two-vacuum-cup suction hand. We propose a deep convolutional neural network (DCNN) trained on 15,000 annotated depth images synthetically generated in a physics simulator [39] to predict grasp points (centers of graspable surfaces) and their grasp patterns, namely, the suction hand vacuum-cup activation mode: left cup on (L), right cup on (R), or both cups on (B). Additionally, we incorporate a surface feature descriptor that extracts surface features (center position and normal) of the surface containing the predicted grasp point, eliminating the need for VS texture features and goal images. Because the surface feature descriptor reconstructs the surface during feature extraction, the predicted grasp point position can be refined, which helps to reduce the influence of unrealistic datasets without using a sim-to-real method.

Our primary contributions are as follows:

- We propose a framework for automatically generating a training dataset in consideration of the grasp pattern for planar-faced object picking in Gazebo [39], which is more efficient and convenient than manual collection.
- We propose a DCNN model to simultaneously predict grasp point positions and their corresponding grasp patterns (vacuum-cup activation modes) for a two-vacuum-cup suction hand.
- We incorporate surface-feature descriptors for DCNN prediction refinement and feature extraction, allowing the system to be free of sim-to-real refinement for DCNN predictions, as well as texture features and goal images for VS.

2. Picking Robot

In this study, we focus on grasping with a two-vacuum-cup suction hand amid a pile of boxes, which are the most common planar-faced objects in a warehouse. As shown in Figure 1a, the picking robot consists of a 6-degree-of-freedom (DoF) manipulator (TVL500; Toshiba Machine Co. Ltd., Shizuoka, Japan) and a 1-DoF suction hand (Figure 1c). The suction hand is an eye-in-hand system having two 0.018-m diameter vacuum cups (Figure 1b), which are symmetrically set across an RGB-D camera (RealSense SR300; Intel, Santa Clara, CA, USA). This camera can capture both RGB color and depth images, but our system uses only the depth images; RGB images are used only to visualize the results.

As shown in Figure 1c, we define the normal, major (long)-axis, and minor (short)-axis directions for the graspable surface of a planar-faced object in the camera frame as \mathbf{n} , \mathbf{n}_l , and \mathbf{n}_s , respectively. L_l and L_s are respectively the long- and short-side lengths of the surface. $\tilde{\mathbf{n}}_l$ and $\tilde{\mathbf{n}}_s$ are respectively the major-axis and minor-axis directions of a warped perspective projected on the graspable surface.

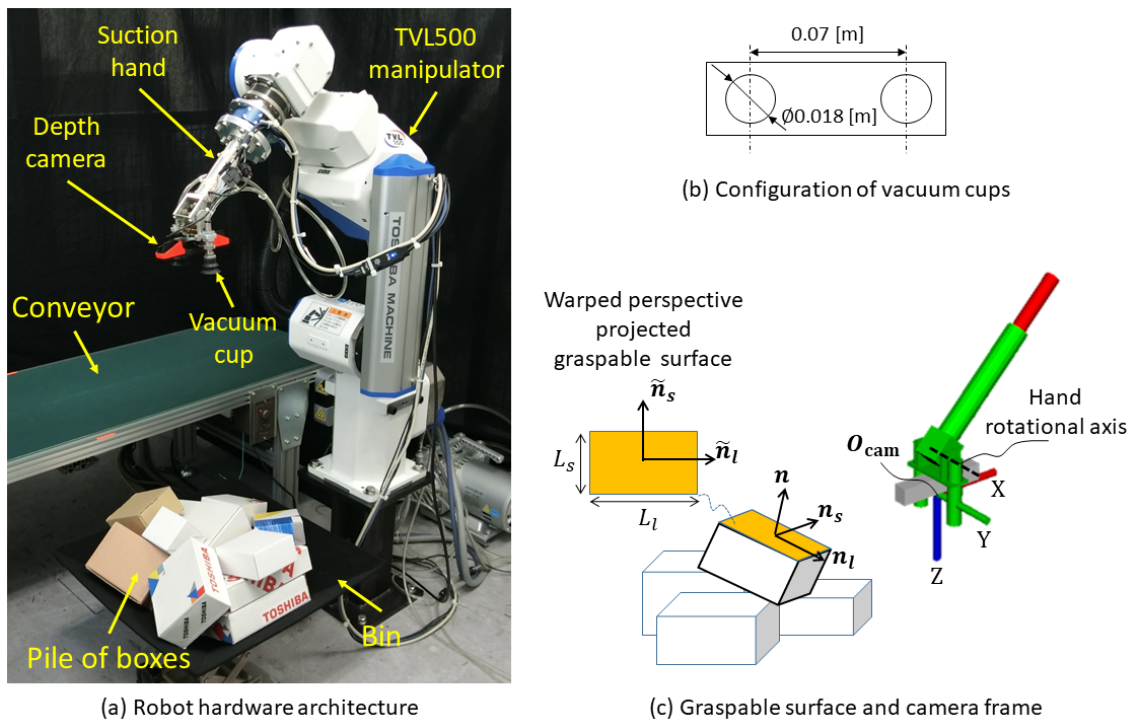


Figure 1. Picking robot. (a) Robot hardware architecture. (b) Configuration of vacuum cups. (c) Graspable surface and camera frame. n , n_l , and n_s respectively indicate the normal, major (long) axis, and minor (short) axis of the graspable surface in the camera frame. L_l and L_s are respectively the long-side and short-side lengths of the graspable surface in the camera frame. \tilde{n}_l and \tilde{n}_s are respectively the major-axis and minor-axis directions of a warped perspective projected on the graspable surface.

3. Vision-Guided Robot Bin-Picking System

As shown in Figure 2, the proposed system comprises a learning-based textureless planar-faced object grasp planning loop (red dashed lines) and a visual guided robot control loop (blue solid lines). Both loops only use depth images as inputs. The grasp planning loop uses a DCNN model trained on a simulated dataset to predict grasp points G (center points of visible surfaces). Unlike previous methods that output grasp point positions only, here G includes a grasp point position g and its corresponding grasp pattern S for determination of vacuum-cup activation. The predicted grasp points are further scored according to distance to the camera and the angle between the point normal at g and camera axis Z . These scores are ranked and the optimal grasp point G_{opt} is sent to the robot control loop. Section 4.2 presents details of the grasp point score ranking.

Given position g_{opt} of G_{opt} , the surface descriptor uses g_{opt} as an initial seeking point to reconstruct the surface including g_{opt} , whose surface features (normal n and center position g'_{opt}) will then be calculated. During surface reconstruction, g_{opt} is refined to g'_{opt} so that even if the predicted g_{opt} is inaccurate, the positional offset error can be compensated for. Based on n , g'_{opt} , and S_{opt} , 2.5D VS calculates reference translational velocity V and rotational velocity Ω of the suction hand for controlling the robot to track and apply suction to the target surface by the predicted vacuum cup activation mode S_{opt} for the optimal grasp point G_{opt} . Note that the grasp planning loop only activates when the robot is at its home position, because the descriptor can use the calculated g'_{opt} from the previous iteration as an initial seeking point for the subsequent iteration.

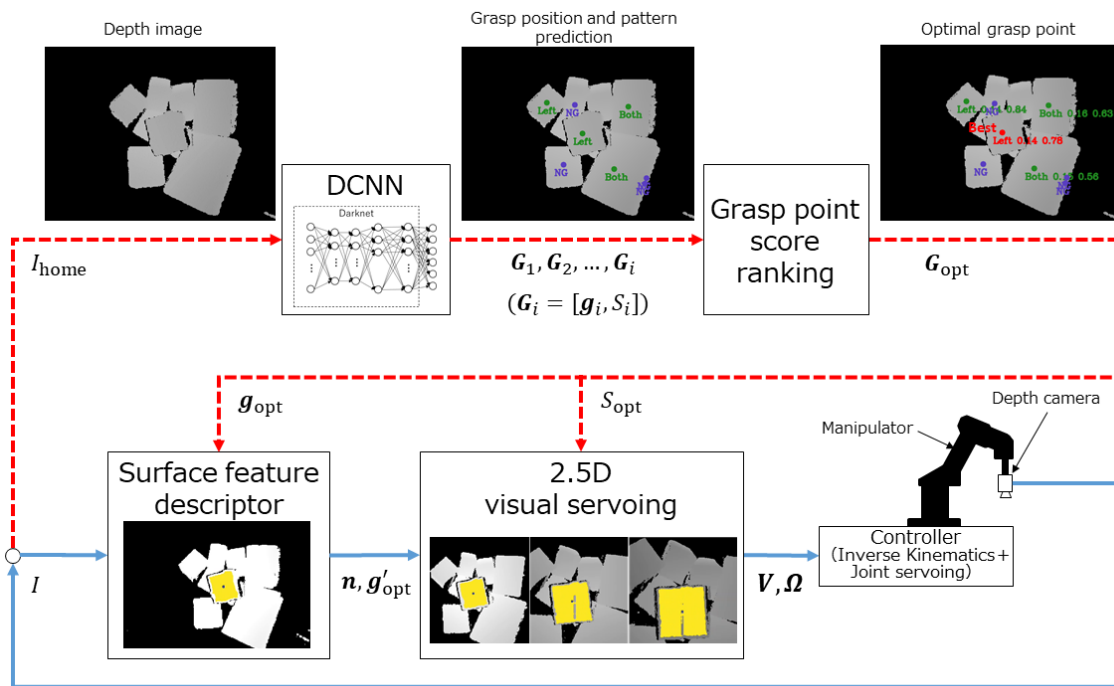


Figure 2. System diagram. The red dashed lines indicate the loop for grasp planning for textureless planar-faced objects by a deep convolutional neural network (DCNN) model at the robot's home position. The blue solid lines indicate the visual guided robot control loop. Both loops only use depth images. I is the depth image, and I_{home} is the depth image captured at the robot's home position. G_i is the i th predicted grasp point, defined as the set of grasp point position g and grasp pattern S . G_{opt} is the optimal grasp point among all predicted points, and g_{opt} and S_{opt} indicate its corresponding position and grasp pattern, respectively. n and g'_{opt} are respectively the surface normal and refined center position, namely the surface feature. V and Ω are respectively the reference translational and rotational velocity of the suction hand.

4. Learning-Based Grasp Planning for Textureless Planar-Faced Objects

The learning-based grasp planning loop for textureless planar-faced objects uses a DCNN model (Section 4.1) to predict G , which includes center points of all visible planar surfaces and their grasp patterns (vacuum cup activation modes) in an input depth image. The predicted points are further ranked and the optimal grasp point is output, as described in Section 4.2. Furthermore, we propose a framework (Section 4.3) for automatically collecting the training dataset for DCNN in a physics simulator.

4.1. DCNN Model

Supervised learning is conducted to directly learn grasp points without segmentation. The DCNN model regresses grasp point position g and predicts the probability of four classes (grasp pattern S) of g , which includes L, R, B, and non-graspable (NG). Here, we consider g as center points of all visible planar surfaces, because the center point of a surface is assumed to be the most stable position to apply suction. Further, the class probability prediction considers the mechanical and kinematic constraints of the robot (e.g., joint limits and inverse kinematics solutions), manipulation efforts, and collision status with the surroundings. Section 4.3 details how we label the class of each g for depth images in the training dataset.

As shown in Figure 3, the DCNN model has 23 convolutional layers and uses Darknet-23 [40] as the backbone. A leaky rectified linear activation function is used for all convolutional layers except the last ("conv. layer*" in Figure 3). DCNN takes a 416×416 depth image as input and predicts a list of

grasping points \mathbf{G} where the maximal probability of the four classes exceeds a threshold. Specifically, \mathbf{G} is defined as

$$\mathbf{G}_i = [\mathbf{g}_i, S_i] \quad (1)$$

$$\mathbf{g}_i = [g_i^u, g_i^v], \quad (2)$$

where

$$S_i = \max(p_R, p_L, p_B, p_{NG}) \quad \text{subject to} \quad S_i > th, \quad (3)$$

and \mathbf{G}_i is i th grasp point. \mathbf{g}_i and S_i are respectively the position and grasp pattern of \mathbf{G}_i . \mathbf{g}_i is represented as the pixel coordinate $[g_i^u, g_i^v]$ in a depth image with horizontal and vertical axes u and v . p_R, p_L, p_B, p_{NG} are probabilities of the R, L, B, and NG classes, respectively. S_i is the maximum of these four classes, subject to the constraint that \mathbf{G}_i will be output only when S_i exceeds the DCNN output threshold th .

In this paper, we focus on random picking amid a pile of objects, so focal loss is used because it addresses class imbalance and exhibits high accuracy in dense objects detection [41]. The DCNN model is trained using the following multipart loss functions:

$$L = \lambda_{coord} L_{coord} + \lambda_{class} L_{class} \quad (4)$$

$$L_{coord} = \frac{1}{B} \sum_{j=1}^{N^2} \mathbb{1}_j^{GP} (g_j^u - \hat{g}_j^u)^2 + (g_j^v - \hat{g}_j^v)^2 \quad (5)$$

$$L_{class} = \frac{1}{B} \sum_{j=1}^{N^2} \sum_{c \in \text{classes}} \begin{cases} -\alpha(1 - p_j(c))^\gamma \log(p_j(c)) & \hat{p}_j(c) = 1 \\ -(1 - \alpha)p_j(c)^\gamma \log(1 - p_j(c)) & \text{otherwise} \end{cases} \quad (6)$$

with L_{coord} evaluating the positional offset by L2 loss and L_{class} evaluating the multiclass classification by focal loss. N ($= 13$) is the grid size of a depth image, which was divided to speed up training. B is the number of ground-truth grasp points in the depth image. g_j^u and g_j^v are pixel coordinates of the grasp point position in the j th cell. c is one of the four classes (R, L, B, or NG). $p_j(c)$ is the probability of c in the j th cell. α and γ are tunable focusing parameters of the focal loss. α and γ are set to 0.25 and 2.0, respectively. λ_{coord} and λ_{class} are penalty weights for positional offset and classification error, respectively. Both λ_{coord} and λ_{class} are set to 1.0. $\mathbb{1}_j^{GP}$ is a flag indicating if there is a ground-truth grasp point in the j th cell; L_{coord} is evaluated only when $\mathbb{1}_j^{GP} = 1$ (a ground-truth grasp point exists).

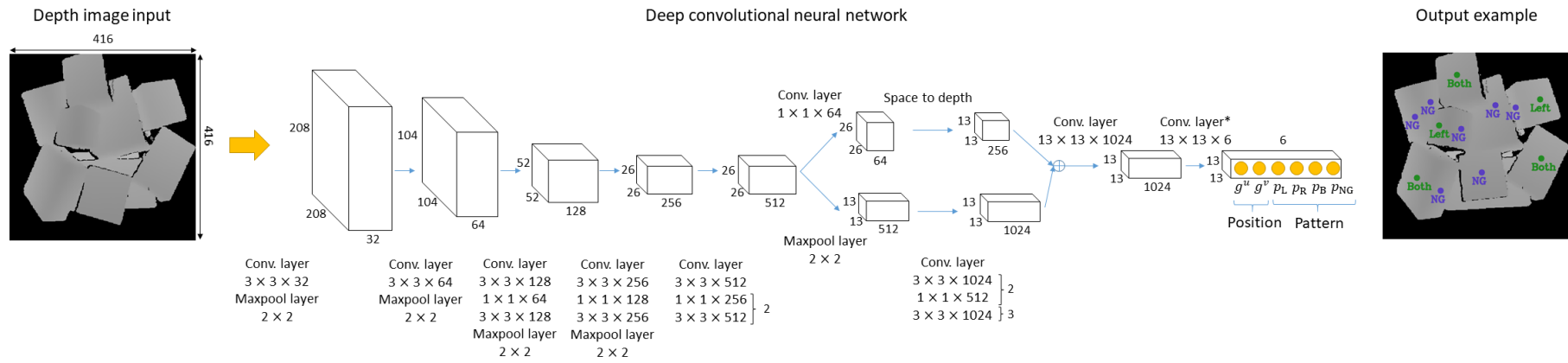


Figure 3. DCNN model.

4.2. Grasp Point Score Ranking

This section describes how predicted grasp points G are scored and the optimal point G_{opt} in G is output. Equation (7) evaluates the graspability score J_i for grasp point i .

We assume that a higher grasp point, in other words one closer to the camera, is faster and safer to access, so J_{distance} (Equation (9)) is defined to evaluate the normalized distance from G_i to the camera center. We also assume that a smaller surface inclination angle is easier to grasp (because a level surface is more easily grasped than a inclined one), so $J_{\text{inclination}}$ (Equation (10)) is defined to evaluate normalized surface inclination angles. ϕ is the angle between the point normal \mathbf{n}_{G_i} at G_i and the camera-axis Z normal \mathbf{n}_Z ($[0, 0, 1]$), as in Equation (8). Further, there is a tradeoff between J_{distance} and $J_{\text{inclination}}$. We prioritize J_{distance} if $\phi < \frac{\pi}{6}$ and $J_{\text{inclination}}$ otherwise, because we empirically found that J_{distance} has a dominant influence on graspability when ϕ is smaller than $\frac{\pi}{6}$, but $J_{\text{inclination}}$ is dominant when ϕ exceeds $\frac{\pi}{6}$.

$$J_i = \begin{cases} J_{\text{distance}} & \phi < \frac{\pi}{6} \\ J_{\text{inclination}} & \text{otherwise} \end{cases} \quad (7)$$

$$\phi = \arccos \frac{\mathbf{n}_{G_i} \cdot \mathbf{n}_Z}{\|\mathbf{n}_{G_i}\| \|\mathbf{n}_Z\|} = \arccos \frac{n_{G_i}^Z}{\|\mathbf{n}_{G_i}\|} \quad (8)$$

$$J_{\text{distance}} = 1 - \frac{I(g_i^u, g_i^v)}{D_{\text{ground}}} \quad (9)$$

$$J_{\text{inclination}} = 1 - \frac{2\phi}{\pi}. \quad (10)$$

Here, \mathbf{n}_{G_i} is the point normal at G_i . $n_{G_i}^Z$ is the Z component of \mathbf{n}_{G_i} , and \mathbf{n}_Z is the camera axis Z normal. $I(g_i^u, g_i^v)$ is the depth at G_i in a depth image, which represents the distance between the camera and G_i along the camera Z axis. D_{ground} is the distance between the ground and the camera at the robot's home position.

4.3. Automatic Training Dataset Collection

Figure 4 shows the automatic training dataset collection framework. As shown in Figure 4a, to generate a pile of textureless planar-faced objects with a physically plausible space configuration, we drop a random number (0 to 20) of boxes in random poses from a height of 0.625 m relative to the bin bottom. To generate a disorganized heap we use boxes of three dimensions, determined based on common warehouse items: $0.12 \times 0.075 \times 0.023$ m, $0.145 \times 0.092 \times 0.052$ m, and $0.06 \times 0.06 \times 0.07$ m. Once the boxes reach a stable state, the camera captures an artificial depth image from a height of 0.625 m above the bin center.

To annotate the generated depth image, the center point position of all visible surfaces is required to annotate the grasp point position. Meanwhile, the grasp pattern selection and a graspability check must be performed to label the class for each grasp point. Given a 2.5 D depth image and camera intrinsic matrix, the point cloud can be simply converted from the depth image. Given that the robotic hand is a vacuum type, we assume the grasp point to be the center of a surface for stable suction. To detect the surface, we use a box filter provided by the Point Cloud Library [42] to conduct point cloud segmentation (Figure 4b). Then, for each segmented point cloud cluster, we use the Random Sample Consensus (RANSAC) [43] algorithm to detect the surface and calculate the surface-center position and normal (Figure 4c). Finally, we convert the positions of surface center points from camera coordinates to depth-image pixel coordinates, which are used as the grasp points.

The four classes include three graspable pattern classes (L, R, and B) and one non-graspable pattern class (NG). We first label the classes for each grasp point (Figure 4d) considering only the suction hand, then check whether the planar-faced object can be grasped by the labeled grasp pattern (Figure 4e) considering the entire robot. Such a two-stage class labeling method is used because

directly conducting robot-level class labeling is time consuming. Through hand-level class labeling, obviously non-graspable points can be quickly found.

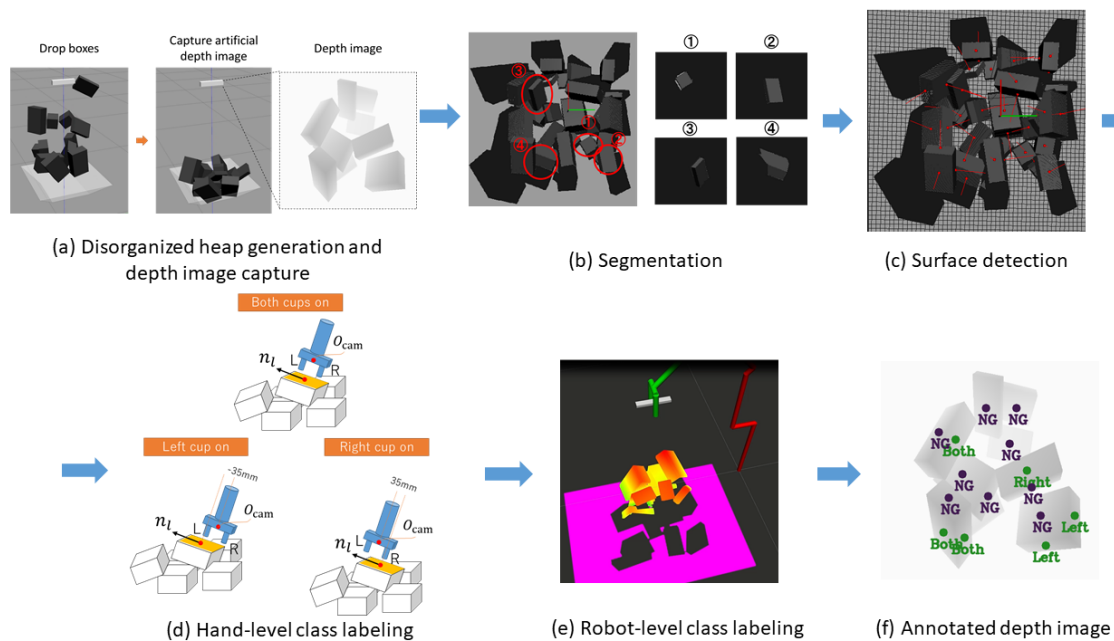


Figure 4. Automatic training dataset collection framework. (a) Disorganized heap generation and depth image capture. (b) Segmentation. (c) Surface detection. (d) Hand-level class labeling. (e) Robot-level class labeling. (f) Annotated depth image.

Figure 5 shows the class labeling algorithm. Hand-level labeling is first conducted based on the depth image. For ease of understanding, Figure 6 shows examples of the hand-level labeling process. Given a target surface depth image, we perform a warped perspective transformation to let the surface face toward the camera, as shown in Figure 6b. The angle θ_1 between the perspective-projected surface long axis \tilde{n}_l and the camera X axis n_x together with angle θ_2 between the surface short axis \tilde{n}_s and camera X axis n_x are calculated. These two angles are calculated to determine the camera rotation pattern shown in Figure 6c. To grasp the object with a smaller manipulation (rotation) effort, the 2D target pose of the camera is aimed at aligning axis X with the surface axis having a smaller axis angle with respect to axis X . The length of the aligned axis is then calculated to check whether the length is sufficiently long to grasp the object with both vacuum cups (Figure 6d). If not, we further check whether one vacuum cup can reach the object. If one cup can reach the surface, we compare performance when using the left and right cups (Figure 6e). The higher camera center O_{cam} , which is closer to the camera center at the robot home position along axis Z , is safer, so the corresponding pattern will be selected as the label. For example, in Figure 6e the left-side O_{cam} is higher from the bin bottom, so the right-cup pattern will be selected. Hand-level labeling can quickly filter out NG grasp points, which are not needed for further robot-level labeling.

Robot-level labeling checks whether a grasp point is graspable by a target hand pose for the entire robot. During hand-level labeling, we determined the 2D target pose for which camera-axis X and Y directions had been calculated. To determine the 6D target pose in the camera frame, O_{cam} and the camera axis Z direction are required. The axis Z direction can be obtained by calculating the anti-direction of the surface norm. Meanwhile, as Figure 4d shows, O_{cam} is a 0.15-m offset from the surface center along the camera axis Z for both cups in the class, while further offsets of 0.035 or -0.035 m are added along camera axis Y for left and right cups, respectively. Note that 0.035 m is the distance from a vacuum cup center to the camera center (see Figure 1b). Given a target suction hand pose, inverse kinematics calculations, a joint-limit check, and a collision check are sequentially

conducted to find plausible joint angles for the robot. If no solution exists, the grasp point is labeled as NG; otherwise the hand-level labeling result is kept.

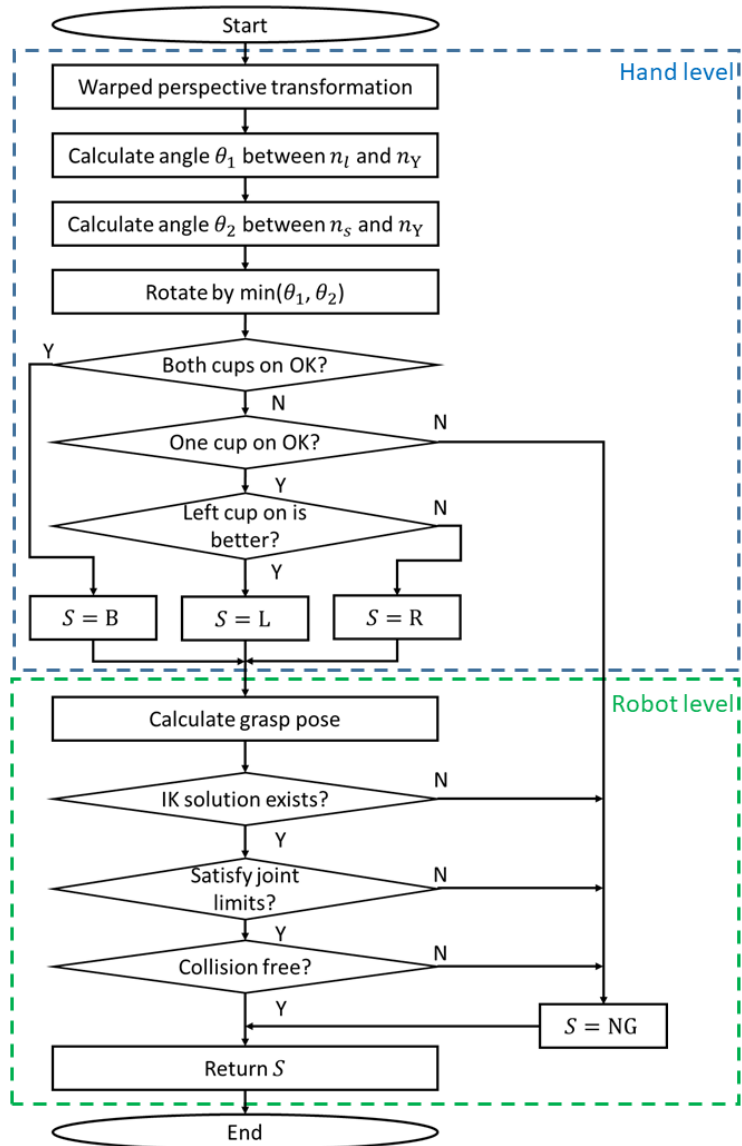


Figure 5. Class-labeling algorithm. The blue dashed line indicates steps for hand-level class labeling. The green dashed line indicates steps for robot-level class labeling. S is a grasp pattern, namely, a class label. L, R, B, and non graspable (NG) indicate left cup on, right cup on, both cups on, and non-graspable, respectively. IK indicates an inverse kinematics calculation.

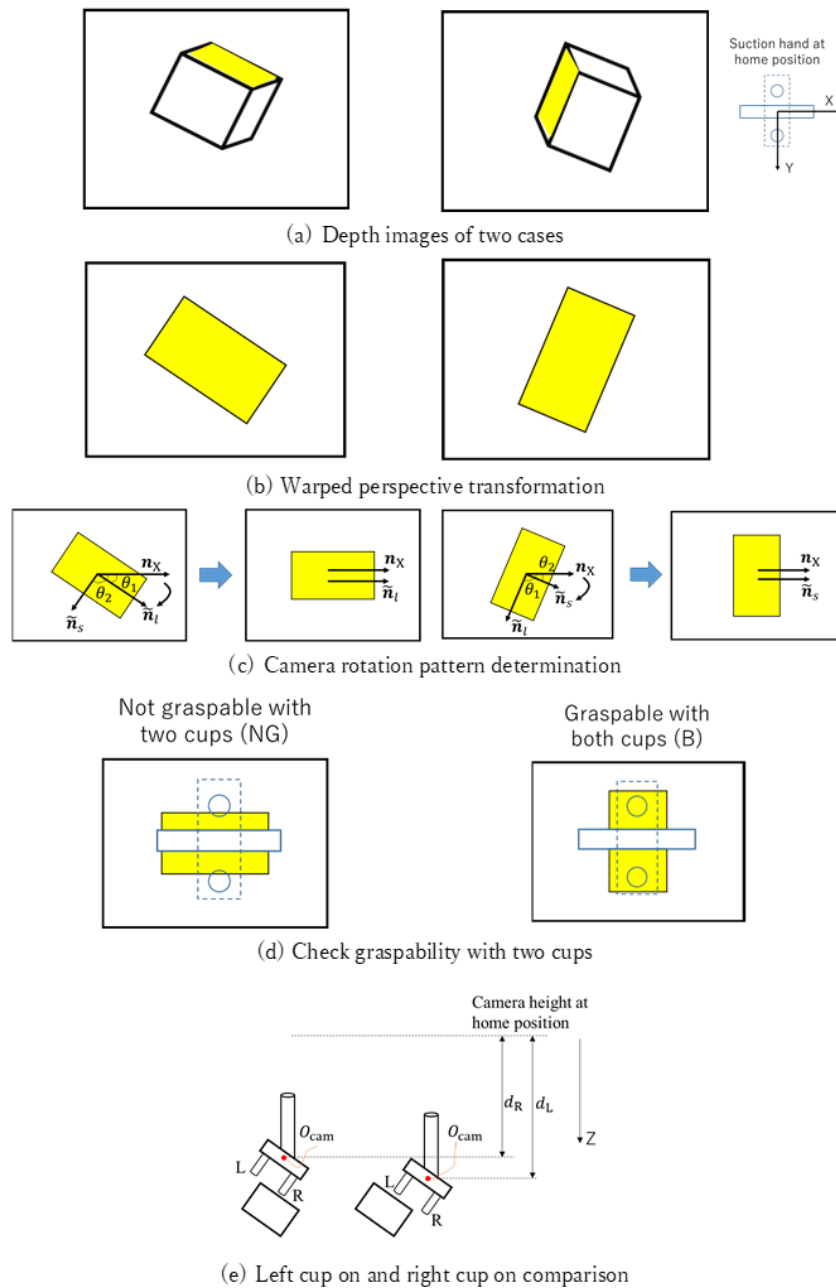


Figure 6. Examples of hand-level class labeling for two depth images. (a) Two cases of depth images. X is the camera long axis. (b) Warped perspective transformation. (c) Camera rotation pattern determination. (d) Check graspability with two cups. (e) Left cup on and right cup on comparison. O_{cam} is the camera center when picking the object. d_R and d_L are the distance between O_{cam} and the camera center at the home position along camera axis Z when picking the object by the corresponding pattern.

5. Visual-Guided Robot Control

As shown in Figure 2, the depth-image-based visual-guided robot control scheme consists of a surface feature descriptor and 2.5D VS. The feature descriptor calculates the center position and normal of the target surface by the region expansion algorithm at each iteration and 2.5D visual servo feedback control is used to calculate the hand velocity for the controller to manipulate the robot to the target pose.

5.1. Surface Feature Descriptor

Figure 7 shows a general framework for the surface feature descriptor. The descriptor performs two functions. The first is to refine g_{opt} prediction by the learning-based grasp planning loop to g'_{opt} , which the surface center g_{opt} belongs to, at the robot's home position in the first iteration. The second is to provide the surface center and normal to the 2.5D VS.

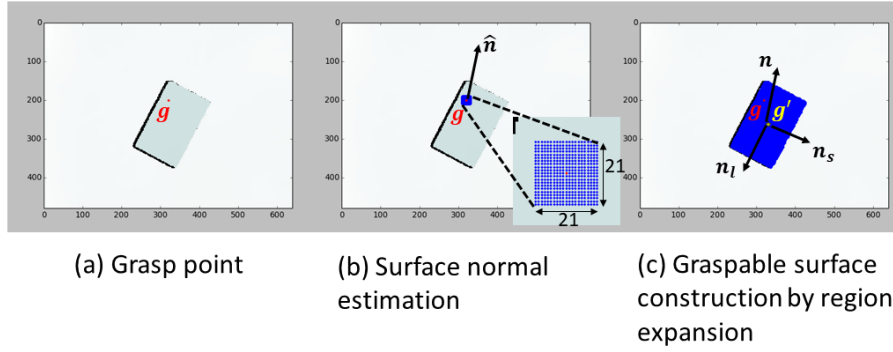


Figure 7. Surface feature descriptor. (a) Initially sought grasp point. The red point g is the point g_{opt} predicted by the learning-based grasp planning loop at the robot home position in the first iteration. From the second iteration, this point will be the calculated surface center g' in the previous iteration. (b) Surface-normal estimation. \hat{n} is the estimated surface normal. (c) Graspable surface reconstruction by region expansion. The blue surface is the constructed graspable surface. The yellow point g' , namely the refined g , is the surface center.

To calculate surface features, we use the region-growing algorithm for surface construction, because region growing is more robust than RANSAC in actual applications [44]. To determine the region expansion rule, an initial seeking point g and the normal of the surface to which g belongs must be calculated. g is g_{opt} at the home position but becomes the calculated g' in the previous iteration after the robot starts to move. To determine the normal, we extract a 21×21 -pixel window (Figure 7b) centered at g , then fit the plane by calculating the singular value decomposition of cloud points in the window. The plane-fitting problem is then to find all points P satisfying

$$\hat{n} \cdot P(u,v) + \hat{D} = 0 \quad (11)$$

in camera coordinates, where \hat{n} is the estimated normal of the fitted plane. $P(u,v)$ is the counterpart of a point (u,v) in a depth image. Namely, $P(u,v)$ is the corresponding point of (u,v) in a point cloud converted from the depth image given the camera-intrinsic matrix.

Then the region expansion rules can be defined to filter out points belonging to the surface (Equation (11)) in the whole depth image as

$$|n_p \cdot P(u,v) + \hat{D}| < \epsilon_1 \quad (12)$$

$$\|n_p - \hat{n}\|^2 < \epsilon_2 \quad (13)$$

$$n_p = (P(u+1,v) - P(u-1,v)) \times (P(u,v+1) - P(u,v-1)), \quad (14)$$

where n_p is the normal of point P .

Given the fitted plane, we can further simply obtain the surface center (refined grasp point) g' , surface long axis n_l , and short axis n_s . We conducted the warped-perspective transformation to project n_l and n_s onto \tilde{n}_l and \tilde{n}_s , which are necessary for VS in Section 5.2.

The surface descriptor finally outputs features including surface normal n in camera coordinates, the warped perspective projected on surface long axis \tilde{n}_l and short axis \tilde{n}_s in camera coordinates, and the surface center g' in pixel coordinates.

5.2. 2.5D Visual Servoing

Velocity control of the gripper in an eye-in-hand system controls the 6 degrees of freedom of the camera. Inspired by 2.5D VS [8], gripper velocity control can be considered as controlling the following feedback error e to be zero:

$$e = \begin{bmatrix} p_e - p_e^* \\ u\theta \end{bmatrix}, \quad (15)$$

where p_e is the normalized g' from the surface feature descriptor and p_e^* is the normalized target position. $u\theta$ gives the angle/axis parametrization for camera rotation.

p_e ($[x, y, z]$) is a perspective projection of g' ($[X, Y, Z]$) to the graspable surface, as in Equation (16). p_e^* is the normalized center of the depth image for class B. This is because all points in the depth image captured at the target pose belong to the graspable surface, and thus the image center point is the target, namely the surface center. For classes L and R, respective offsets of -0.035 and 0.035 m are added as follows:

$$p_e = [x, y, z]^T = \left[\frac{X}{Z}, \frac{Y}{Z}, \log(Z) \right]^T \quad (16)$$

$$p_e^* = \left[0, \frac{off}{d^*}, \log(d^*) \right]^T \quad (17)$$

$$off = \begin{cases} -0.035 & S = L \\ 0 & S = B \\ 0.035 & S = R \end{cases} \quad (18)$$

where d^* is the distance between the camera frame and the graspable surface at the target pose. Specifically, d^* indicates the offset distance from g' along the n direction.

We use $\dot{e} = -\lambda e$ to make $p_e - p_e^*$ and $u\theta$ in Equation (15) converge to zero. As reported in [5,6], \dot{e} can be represented as

$$\dot{e} = \begin{bmatrix} \dot{p}_e \\ u\dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{1}{d^*\rho} L_v & L_w \\ \mathbf{0} & I_3 \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix} \quad (19)$$

$$L_v = \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1 \end{bmatrix} \quad (20)$$

$$L_w = \begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \\ -y & x & 0 \end{bmatrix} \quad (21)$$

$$\rho = \frac{Z}{d^*}, \quad (22)$$

where V and Ω are respectively the translational and rotational velocities of the camera. The gripper velocity control law $\dot{e} = -\lambda e$ can thus be described as

$$\begin{bmatrix} V \\ \Omega \end{bmatrix} = -\lambda \begin{bmatrix} \frac{1}{d^*\rho} L_v & L_w \\ \mathbf{0} & I_3 \end{bmatrix}^{-1} \begin{bmatrix} p_e - p_e^* \\ u\theta \end{bmatrix}, \quad (23)$$

where λ is the convergence rate, which is set to 0.4.

To let camera axis Z coincide with graspable surface normal \mathbf{n} and camera axis X coincide with axis $(\tilde{\mathbf{n}}_l$ or $\tilde{\mathbf{n}}_s)$ with smaller rotational effort, we make $\mathbf{u}\theta$ converge to zero. Assuming infinitesimal rotations of those axes, $\mathbf{u}\theta$ can be approximately represented as

$$\mathbf{u}\theta = (\mathbf{n} \times \mathbf{n}_Z)\theta_{\mathbf{n},\mathbf{n}_Z} + \begin{cases} (\tilde{\mathbf{n}}_l \times \mathbf{n}_X)\theta_{\tilde{\mathbf{n}}_l,\mathbf{n}_X} & \theta_{\tilde{\mathbf{n}}_l,\mathbf{n}_X} < \theta_{\tilde{\mathbf{n}}_s,\mathbf{n}_X} \\ (\tilde{\mathbf{n}}_s \times \mathbf{n}_X)\theta_{\tilde{\mathbf{n}}_s,\mathbf{n}_X} & \text{otherwise} \end{cases} \quad (24)$$

$$\theta_{\mathbf{n},\mathbf{n}_Z} = \arccos\left(\frac{\mathbf{n} \cdot \mathbf{n}_Z}{\|\mathbf{n}\|\|\mathbf{n}_Z\|}\right) \quad (25)$$

$$\theta_{\tilde{\mathbf{n}}_l,\mathbf{n}_X} = \arccos\left(\frac{\tilde{\mathbf{n}}_l \cdot \mathbf{n}_X}{\|\tilde{\mathbf{n}}_l\|\|\mathbf{n}_X\|}\right) \quad (26)$$

$$\theta_{\tilde{\mathbf{n}}_s,\mathbf{n}_X} = \arccos\left(\frac{\tilde{\mathbf{n}}_s \cdot \mathbf{n}_X}{\|\tilde{\mathbf{n}}_s\|\|\mathbf{n}_X\|}\right), \quad (27)$$

where \mathbf{n}_X and \mathbf{n}_Z are the normals of camera axes X $([1, 0, 0])$ and Z $([0, 0, 1])$, respectively. The first term in Equation (24) aligns camera axis Z with the surface normal and the second term aligns camera axis X with the long or short axis having the smaller angle respective to axis X .

Thus, the gripper-reference translational and rotational velocities can be calculated based on the depth image and a camera-intrinsic parameter matrix. In Equation (23), d^* is set to 0.15 m, indicating that the gripper is sufficiently close to the graspable surface for grasping. ρ can be calculated based on Equation (22), where, given the camera-intrinsic parameter matrix, Z can be derived from the depth image at a certain point.

The joint reference angular velocities \mathbf{q}_r of the robot can thus be simply calculated as

$$\mathbf{q}_r = \int \dot{\mathbf{q}}_r dt = \int J_c^{-1} \begin{bmatrix} \mathbf{V} \\ \boldsymbol{\Omega} \end{bmatrix} dt, \quad (28)$$

where J_c^{-1} is the pseudoinverse of the Jacobian matrix in camera coordinates.

6. Experiment

To train the DCNN model, we generated 15,000 annotated depth images of a pile of boxes in Gazebo using the framework proposed in Section 4.3. We then conducted experiments of DCNN training based on the artificial dataset. Experiments were conducted on a computer with an Intel Core i7-6700K CPU operating at 4.00 GHz and an Nvidia Geforce GTX 1080Ti GPU. The DCNN was implemented in Keras. The Adam optimizer was used with its suggested default parameters ($\alpha = 0.0001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$). The batch size was set to 16. The precision was calculated to evaluate the DCNN performance.

To validate grasp planning performance, we compared performance for grasp point position predictions with Dex-Net 4.0, which uses only depth images as input. We evaluated performance according to percentage of predicted optimal grasp points with a nearest distance to ground truth surface center points of less than 30 pixels.

To validate efficiency of the proposed robot bin-picking system, experiments involved pick-and-place tasks in which robots picked disorganized planar-faced objects in a bin and placed them on a conveyor, as in Figure 1. The target planar-faced objects were a mix of textureless and textured boxes. As Figure 8 shows, boxes with varying sizes were used in the experiment, including 5 textured and 7 textureless boxes.

Twelve boxes were picked in each of 10 trials. Only a depth image captured by an Intel SR300 camera was used. In each trial, the robot performed grasp attempts until all boxes are removed from the bin. The task was considered complete when no objects remained in the bin, or as failure after 5 unsuccessful grasp attempts for a target box or when the grasp planning loop failed to predict a

grasp point at the robot's home position. The grasp planning loop ran on a computer with an Intel Core i7-4790 CPU operating at 3.60 GHz and an Nvidia GeForce 1080 GPU, while the VS loop ran on a computer with an Intel Core i7-9700K CPU operating at 3.60 GHz. These computers communicated using an ROS (an open-source robotic operating system) communication protocol [45]. The average success rate of ten trials and takt time (pieces per hour (PPH) with successful grasps) were calculated and compared with previous studies.

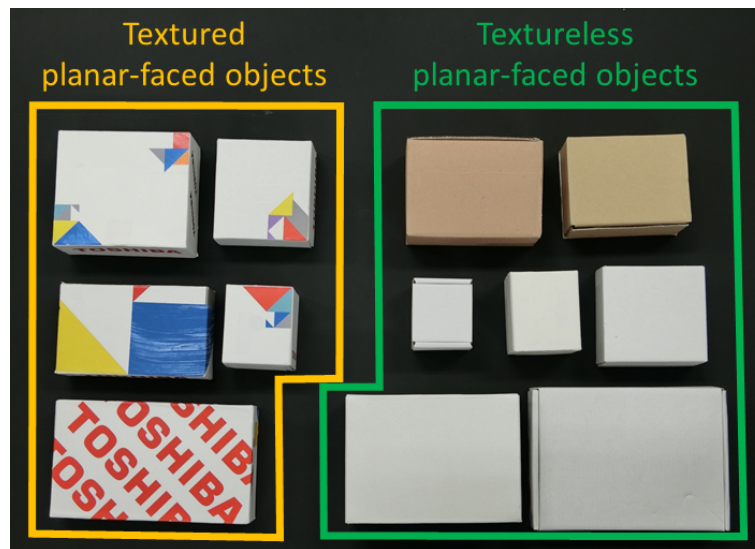


Figure 8. The object set. The yellow region indicates textured boxes. The green region indicates textureless boxes.

7. Results

Table 1 shows the precision of DCNN model prediction. Precision for the top 1 and top 1% predictions exceed 80%, which can be considered as a reliable level of output.

Table 2 shows percentages of grasp points with distance to a surface center of less than 30 pixels. Both Dex-Net and our grasp planning method can predict surface grasp points, but our method is more likely to predict ones close to the surface center (79.16% for our method vs. 58.88% for Dex-Net).

Table 1. Precision of DCNN model prediction.

Top 1	Top 1%	Top 5%	Top 10%
83.10%	80.25%	67.22%	55.64%

Table 2. Percentage of grasp points with a distance to a surface center of less than 30 pixels.

Dex-Net 4.0	Ours
58.88%	79.16%

Further, our system achieved a 97.5% success rate at a speed exceeding 1000 PPH, as demonstrated in the video (Supplementary File 1) in Supplementary Material. Both textured and textureless boxes could be held by a suction hand. Figure 9 shows examples of successful grasp attempts. The grasp planning loop successfully output the best grasp point with its position and grasp pattern at the robot home position. The VS loop then started to track and access the surface containing the point. Although the DCNN model was trained on a synthesized depth image, refinement of the DCNN prediction by the surface descriptor makes the picking system efficient. Table 3 compares average success rates and takt times with those of previous studies.

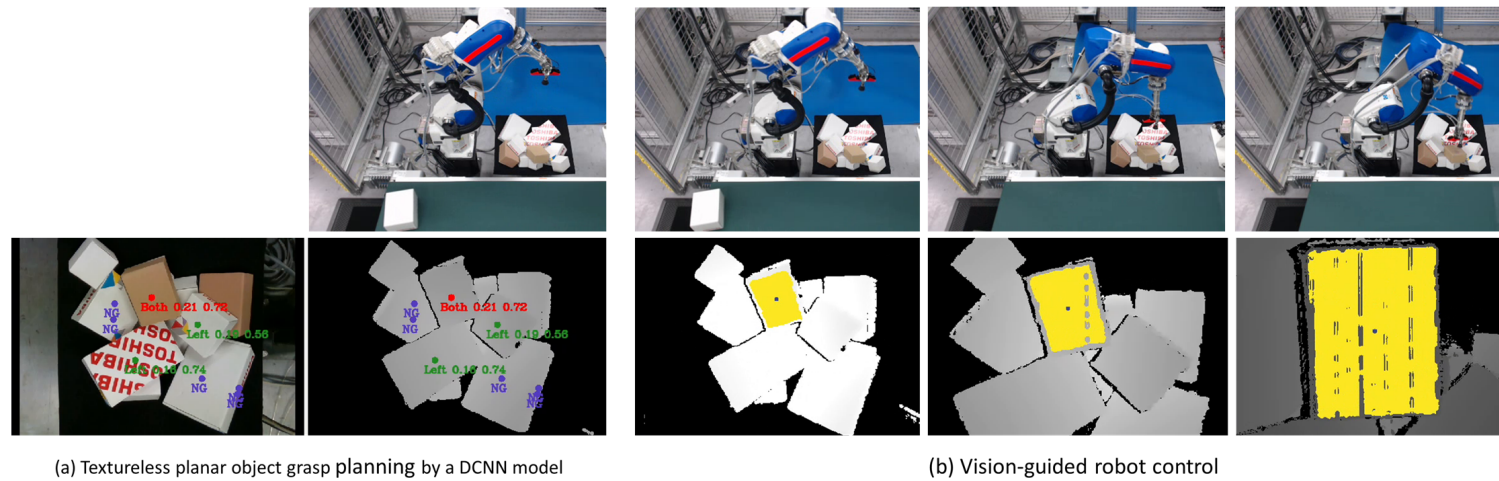


Figure 9. Example of one grasp attempt. (a) Grasp planning for textureless planar-faced objects by a DCNN model. Note that RGB color is only for visualization of the results. Text in grasp point labels indicates the grasp pattern class. First and second values beside the text are J_{distance} and $J_{\text{inclination}}$, respectively. (b) Vision-guided robot control. Yellow areas are target surfaces and blue points are surface centers.

Table 3. Comparison of bin-picking system performance.

System	Object Set	Robotic Hand	Input	Success Rate (%)	grasp Planning Time Cost (s)	Mean Piece per Hour (PPH)
Dex-Net 3.0 [33]	Prismatic solids	Suction (1 cup)	Depth	98	About 3.0	-
Dex-Net 4.0 [46]	Level 1 objects	Suction (1 cup) + gripper (2 fingers)	Depth	97 *	-	Over 300
Zeng et al. [47]	APC dataset	Suction (1 cup) + gripper (2 fingers)	RGB + depth	58.3	0.06	-
Le et al. [48]	USB flash drive packs	Suction (1 cup)	RGB + point cloud	99.6	0.862	-
Ours	Boxes	Suction (2 cups)	Depth	97.5	0.034	Over 1000

Level 1 objects consist of prismatic and circular solids (e.g., boxes, cylinders) common in groceries, toys, and medicine. The ARC dataset indicates the Amazon Picking Challenge (ARC) dataset. "*" indicates the success rate and time cost of grasping by a gripper/suction cup hybrid.

8. Discussion

This study aimed to propose a novel depth image-based vision-guided robotic bin-picking system for textureless planar-faced objects utilizing a two-vacuum-cup suction hand. To the best of our knowledge, no previous studies have proposed a DCNN model for predicting grasp patterns (vacuum cup activation modes) for a two-vacuum-cup suction hand.

Experimental results demonstrated that the proposed bin-picking system can efficiently complete pick-and-place tasks for textureless planar-faced objects with a 97.5% success rate at speeds exceeding 1000 PPH. The DCNN model can predict the grasp point position and its corresponding proper grasp pattern. A surface feature descriptor further refines DCNN output and calculates surface features to make the VS free from texture features.

Only three box types were used during DCNN model training, but the robot could pick all 12 box types in the experiment, indicating that a DCNN model might be able to predict grasp points for novel boxes because they have similar convolutional features.

8.1. Comparison with Previously Proposed Bin-Picking Systems

The proposed system has been compared with previously proposed bin-picking systems [33,46–48] by grasp planning performance and overall system efficiency.

8.1.1. Grasp Planning Performance

We compared grasp planning performance using the proposed method with that using Dex-Net 4.0 [46], because both systems use only depth images. Both output grasp point positions (though Dex-Net does not output grasp patterns), so we evaluated performance according to the percentage of predicted optimal grasp points with a nearest distance to a ground-truth surface center point of less than 30 pixels. The results indicate that our method is more likely to predict a grasp point close to the surface center (see output examples of the two systems in Figure 10). These results agree with previous results [47] showing that Dex-Net may perform poorly at predicting grasp points near the center of mass of an object. Note that we did not perform a comparison with [47,48] because color images are indispensable for the grasp planning method described there.

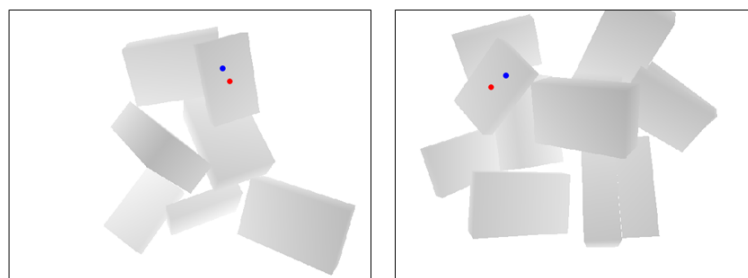


Figure 10. Planned optimal grasp point. The red and blue points are our grasp planning method and Dex-Net 4.0 outputs, respectively.

8.1.2. Overall System Efficiency

We compared the efficiency of our system with data reported in [33,46–48]. Our system outperforms previously proposed bin-picking systems [33,46–48] in terms of grasp planning processing times and mean PPH. This comparison is reasonable despite the different object sets for two reasons. First, the time cost of grasp planning depends on only the input image size and CNN model complexity, so it will not change with object shape complexity. Also, those systems would likely have improved success rates if they were applied to box-picking tasks, because their object sets include objects with complicated shapes and success rates may increase when picking objects with simple shapes. However, we assume that success rate levels of 95% or higher are acceptable for bin-picking of common objects (e.g., boxes) in a warehouse. The main advantage of our system is that it can achieve high success

rates at high speed (1000 PPH, which is approximately three times faster than Dex-Net) for picking planar-faced objects with rectangular surface contours.

Compared with Dex-Net [33,46], our system achieves the same level of success rates at higher speeds. Dex-Net 4.0 trains a GQ-CNN to predict grasp probabilities of grasp candidates for a one-cup-suction hand and a two-finger-gripper. Then, the highest-quality grasp candidate and its corresponding robotic arm are selected to grasp the object. In such multifunction robotic systems and picking methods (grasping or suction), selection policy learning allows manipulation of objects with complex shapes. However, Dex-Net 3.0 and Dex-Net 4.0 use a cross-entropy method to iteratively search over the policy action space for the best grasp, which may result in higher computation costs than that of our one-shot grasp detection method.

Zeng et al. [47] also used a multifunction robotic hand, defining four motion primitives and training a multimodal CNN to predict grasp affordances for each. However, their system requires both RGB and depth images and preparation of grasp affordance heatmaps for predefined motion primitives, which might be difficult in a warehouse. We suggest that when focusing on textureless planar-faced objects in warehouse picking tasks, learning the grasp points of graspable surfaces rather than pixel-wise learning of an affordable map is sufficient. Training dataset preparation will therefore become easier because pixel-wise labeling is unnecessary.

Another recent study [48] proposed a planar-faced object bin-picking system by a two-stage grasp planning method, conducting experiments on USB flash drive packs as representative thin planar objects. They first used a CNN to segment the packs and classify their front and back sides, then estimated object poses to plan a grasp. They achieved a very high success rate (99.6%), but required a longer processing time (0.9 s) than ours. The probable reason for higher processing speeds in our system is that it directly detects grasp points without segmentation. Further, that study only dealt with one type of planar object (the same USB flash drive pack), while our system can grasp planar-faced objects with varying size.

Further, those systems [33,46–48] used one-cup suction hands, while our system uses a two-vacuum-cup suction hand and learned appropriate activation policies. For objects with larger planar surfaces in particular, suction by two cups is obviously more stable than that by only one cup.

8.2. Benefits of Bin-Picking Policies Considering Grasp Pattern Prediction

An efficient learning-based bin-picking system aims at finding a policy that maximizes PPH. Mahler et al. [46] regarded this as a sequential learning problem in which grasp actions affect future states of the heap, and suggested that the performance of a policy trained by supervised learning is sufficient for sequential picking tasks, as compared with imitation learning and reinforcement learning. Our results agree with their idea, in that a CNN for grasp point position and grasp pattern prediction ranking along with a grasp point ranking process can achieve high efficiency.

The conventional bin picking policy [46] considers collision situations and force analytics. Our method does not consider analytic force information based on the assumption that surface area larger than that of the vacuum cup are considered as graspable. In addition to the collision situation, our predicted grasp pattern also considers robot rotation efforts and camera-center position height. Incorporating these factors improves the efficiency of suction by two vacuum cups. For example, lower rotation efforts with respect to the robot pose at its home position helps the robot access the surface more quickly. Considering camera center height helps the robot to select a grasp pattern with lower collision risk.

8.3. Benefits of Depth-Image-Based Visual-Guided Bin-Picking System

One benefit of a depth-image-based visually guided bin-picking system is that it does not require texture features, which are sensitive to environmental brightness; objects imaged under poor lighting conditions may reduce object tracking capabilities by template matching [49] or object recognition by a

CNN [50]. Depth features are more robust, because they can be measured by an infrared camera in dark environments.

Another benefit is that depth images are easier to synthesize than color images. For synthesized color images, sim-to-real technologies such as domain randomization are required to improve performance [51]. Our results demonstrated that a DCNN model trained on synthesized depth images can be directly applied to box-bin picking with high performance, given a properly designed descriptor for DCNN model output refinement.

8.4. Grasp Failure Analysis

Grasp attempts failed in cases where the targeted box was adjacent to another box as shown in Figure 11, where the expected target object is the upper box but the robot targets the wrong box because its surface inclination angle is smaller than that of the expected target. Our policy (Equation 7) prioritizes surface inclinations with angles exceeding 30 degrees and ignores the evaluation of surface height. One method to resolve this problem would be to additionally evaluate differences between surface inclination angles, conducting further height evaluations if the difference is smaller than some threshold. We will thus modify this policy in the future work. Failures also occurred when surface areas were close to that of the vacuum cup, possibly causing the cup to only partly contact the surface and thus lowering the suction force to below expectations. One solution to this problem would be to add a safety constraint on the surface area, stipulating that, for example, the target surface area must be at least 1.2 times larger than the cup area.

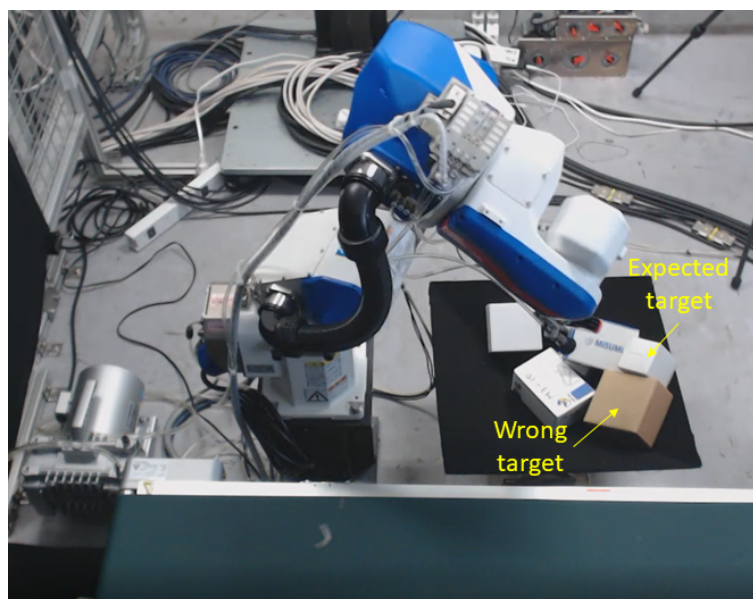


Figure 11. Grasp failure example.

8.5. Limitations and Future Work

This study has several limitations to address in future work. Our picking system is designed for only random bin-picking tasks in which the system cannot grasp a specific object, because we directly detect grasp points without object segmentation and detection. Object segmentation might be required for certain tasks such as picking a specific item ordered by a customer.

In addition, the current implementation can deal with only planar-faced objects with rectangular contours, because the training dataset was generated using only boxes where the lengths of long and short surface sides are required for class labeling (Figure 6). We will improve the class-labeling framework to allow for other objects with planar surfaces but nonrectangular contours, such as circles or polygons, by calculating a bounding box for surface contours and utilizing its size information. Moreover, we will generate the training dataset in a simulator by using various planar-faced objects,

including ones with complicated shapes rather than only boxes. However, increased shape complexity may decrease DCNN prediction accuracy, so to address this issue we will utilize more complicated backbones such as ResNet-101 and expand the dataset volume.

Furthermore, sensory noise (e.g., camera depth image noise) and environmental uncertainty may affect accuracy of the DCNN prediction results and the robustness of the visual servoing control scheme. In this study, we used only a post-processing-filter provided by the Intel RealSense SDK, and the experimental results (97.5% grasp success rate) suggest that such image preprocessing is effective. Employing an information fusion process would be one effective way of further improving the system robustness. Recent studies have shown that Kalman filters and particle filters greatly contribute to human hand tracking by compensating for tracking error and sensor noise [52–54]. The current picking system uses only one robotic-hand-mounted camera. Using extra fixed cameras and integrating multicamera sensory information via a Kalman filter may improve the system robustness to noise and occlusion. Another possibility is to adopt a sliding mode control to improve object tracking robustness while considering unknown disturbances and uncertainties [55].

Another limitation is that experimental conditions such as the relative locations of robots and bins may affect bin-picking task efficiency. Future work will therefore investigate the efficiency of the system under different experimental conditions.

Supplementary Materials: The following are available at <http://www.mdpi.com/1424-8220/20/3/706/s1>.

Author Contributions: Conceptualization, P.J., Y.I., N.S., J.O., S.T., A.S. and A.O.; methodology, P.J., Y.I., N.S., J.O., S.T., A.S. and A.O.; software, P.J., Y.I. and N.S.; validation, P.J., Y.I., N.S. and J.O.; formal analysis, P.J. and Y.I.; data curation, P.J.; writing—original draft preparation, P.J. and Y.I.; writing—review and editing, P.J. and Y.I.; visualization, P.J., A.O., S.T. and A.O.; supervision, P.J., Y.I. and A.O.; project administration, A.O. All authors have read and agree to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Yamaguchi, T. Japan's robotic future. *Crit. Asian Stud.* **2019**, *51*, 134–140. [[CrossRef](#)]
2. Janabi-Sharifi, F. Visual Servoing: Theory and applications. In *Opto-Mechatronic Systems Handbook*; Cho, H., Ed.; CRC: Boca Raton, FL, USA, 2002; pp. 15–1–15–24.
3. Wilson, W.J.; Hulls, C.W.; Bell, G.S. Relative end-effector control using cartesian position based visual servoing. *IEEE Trans. Robot. Autom.* **1996**, *12*, 684–696. [[CrossRef](#)]
4. Thuilot, B.; Martinet, P.; Cordesses, L.; Gallice, J. Position based visual servoing: Keeping the object in the field of vision. In Proceedings of the 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292), Washington, DC, USA, 11–15 May 2002; Volume 2, pp. 1624–1629.
5. Malis, E.; Chaumette, F.; Boudet, S. 2 1/2 D visual servoing. *IEEE Trans. Robot. Autom.* **1999**, *15*, 238–250. [[CrossRef](#)]
6. Malis, E.; Chaumette, F. 2 1/2 D visual servoing with respect to unknown objects through a new estimation scheme of camera displacement. *Int. J. Comput. Vis.* **2000**, *37*, 79–97. [[CrossRef](#)]
7. Malis, E.; Chaumette, F. Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods. *IEEE Trans. Robot. Autom.* **2002**, *18*, 176–186. [[CrossRef](#)]
8. Horaud, R.; Dornaika, F.; Espiau, B. Visually guided object grasping. *IEEE Trans. Robot. Autom.* **1998**, *14*, 525–532. [[CrossRef](#)]
9. Mahler, J.; Goldberg, K. Learning deep policies for robot bin picking by simulating robust grasping sequences. *Proc. Mach. Learn. Res.* **2017**, *78*, 515–524.
10. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [[CrossRef](#)]
11. Suga, A.; Fukuda, K.; Takiguchi, T.; Arika, Y. Object recognition and segmentation using SIFT and Graph Cuts. In Proceedings of the 2008 19th International Conference on Pattern Recognition, Tampa, FL, USA, 8–11 Dec 2008; pp. 1–4.

12. 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. 2961–2969.
13. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 21–37.
14. Redmon, J.; Farhadi, A. YOLO9000: better, faster, stronger. *arXiv* **2017**, arXiv:1612.08242.
15. Badrinarayanan, V.; Kendall, A.; Cipolla, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [[CrossRef](#)]
16. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
17. Li, Y.; Bu, R.; Sun, M.; Chen, B. PointCNN. *arXiv* **2018**, arXiv:1801.07791.
18. Zhikai, D.; Sicheng, L.; Tao, Z.; Hui, C.; Long, Z.; Xingyao, Y.; Houde, L. PPR-Net: Point-wise Pose Regression Network for Instance Segmentation and 6D Pose Estimation in Bin-picking Scenarios. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 4–8 November 2019; pp. 1773–1780.
19. Drost, B.; Ulrich, M.; Navab, N.; Ilic, S. Model globally, match locally: Efficient and robust 3D object recognition. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 998–1005.
20. Kruglyak, L.; Lander, E.S. Complete multipoint sib-pair analysis of qualitative and quantitative traits. *Am. J. Hum. Genet.* **1995**, *57*, 439. [[PubMed](#)]
21. Vidal, J.; Lin, C.Y.; Martí, R. 6D pose estimation using an improved method based on point pair features. In Proceedings of the 2018 4th International Conference on Control, Automation and Robotics (ICCAR), Auckland, New Zealand, 20–23 April 2018; pp. 405–409.
22. Xiang, Y.; Schmidt, T.; Narayanan, V.; Fox, D. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv* **2017**, arXiv:1711.00199.
23. Sock, J.; Kim, K.I.; Sahin, C.; Kim, T.K. Multi-task deep networks for depth-based 6D object pose and joint registration in crowd scenarios. *arXiv* **2018**, arXiv:1806.03891.
24. Zeng, A.; Yu, K.T.; Song, S.; Suo, D.; Walker, E.; Rodriguez, A.; Xiao, J. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1386–1383.
25. Sahbani, A.; El-Khoury, S.; Bidaud, P. An overview of 3D object grasp synthesis algorithms. *Robot. Auton. Syst.* **2012**, *60*, 326–336. [[CrossRef](#)]
26. Bicchi, A.; Kumar, V. Robotic grasping and contact: A review. In Proceedings of the 2000 IEEE International Conference on Robotics and Automation (Cat. No. 00CH37065), San Francisco, CA, USA, 24–28 April 2000; Volume 1, pp. 348–353.
27. Caldera, S.; Rassau, A.; Chai, D. Review of Deep Learning Methods in Robotic Grasp Detection. *Multimodal Technol. Interact.* **2018**, *2*, 57. [[CrossRef](#)]
28. Kumra, S.; Kanan, C. Robotic grasp detection using deep convolutional neural networks. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 769–776.
29. Levine, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Robot. Res.* **2018**, *37*, 421–436. [[CrossRef](#)]
30. Tobin, J.; Biewald, L.; Duan, R.; Andrychowicz, M.; Handa, A.; Kumar, V.; McGrew, B.; Ray, A.; Schneider, J.; Welinder, P.; et al. Domain randomization and generative models for robotic grasping. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 3482–3489.
31. Bousmalis, K.; Irpan, A.; Wohlhart, P.; Bai, Y.; Kelcey, M.; Kalakrishnan, M.; Downs, L.; Ibarz, J.; Pastor, P.; Konolige, K.; et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 4243–4250.

32. 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.
33. Mahler, J.; Matl, M.; Liu, X.; Li, A.; Gealy, D.; Goldberg, K. Dex-Net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 1–8.
34. Gariépy, A.; Ruel, J.C.; Chaib-draa, B.; Giguère, P. GQ-STN: Optimizing One-Shot Grasp Detection based on Robustness Classifier. *arXiv* **2019**, arXiv:1903.02489.
35. Viereck, U.; Pas, A.t.; Saenko, K.; Platt, R. Learning a visuomotor controller for real world robotic grasping using simulated depth images. *arXiv* **2017**, arXiv:1706.04652.
36. Morrison, D.; Corke, P.; Leitner, J. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *arXiv* **2018**, arXiv:1804.05172.
37. 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. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 1–8.
38. Johns, E.; Leutenegger, S.; Davison, A.J. Deep learning a grasp function for grasping under gripper pose uncertainty. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, South Korea, 9–14 October 2016; pp. 4461–4468.
39. Koenig, N.P.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2149–2154.
40. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, CA, USA, 26 June–1 July 2016; pp. 779–788.
41. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
42. Rusu, R.B.; Cousins, S. 3D is here: Point cloud library (pcl). In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 1–4.
43. Yang, M.Y.; Förstner, W. Plane detection in point cloud data. In Proceedings of the 2nd International Conference on Machine Control Guidance, Bonn, Germany, 9–11 Mar 2010; Volume 1, pp. 95–104.
44. Zhang, Z.; Huang, Y.; Zhang, W.; Luo, J. Comparisons of planar detection for service robot with RANSAC and region growing algorithm. In Proceedings of the 2017 36th Chinese Control Conference (CCC), Dalian, China, 26–28 July 2017; pp. 11092–11097.
45. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.
46. Mahler, J.; Matl, M.; Satish, V.; Danielczuk, M.; DeRose, B.; McKinley, S.; Goldberg, K. Learning ambidextrous robot grasping policies. *Sci. Robot.* **2019**, *4*, eaau4984. [[CrossRef](#)]
47. 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. Robot. Res.* **2019**, doi:10.1177/0278364919868017. [[CrossRef](#)]
48. Tuan-Tang, L.; Chyi-Yeu, L. Bin-Picking for Planar Objects Based on a Deep Learning Network: A Case Study of USB Packs. *Sensors* **2019**, *19*, 3602.
49. Talmi, I.; Mechrez, R.; Zelnik-Manor, L. Template matching with deformable diversity similarity. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 175–183.
50. Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; Song, D. Robust physical-world attacks on deep learning models. *arXiv* **2017**, arXiv:1707.08945.
51. Zhang, F.; Leitner, J.; Ge, Z.; Milford, M.; Corke, P. Adversarial discriminative sim-to-real transfer of visuo-motor policies. *Int. J. Robot. Res.* **2019**, *38*, 1229–1245. [[CrossRef](#)]
52. Du, G.; Zhang, P.; Li, D. Human–manipulator interface based on multisensory process via Kalman filters. *IEEE Trans. Ind. Electron.* **2014**, *61*, 5411–5418.

53. Du, G.; Zhang, P. A markerless human–robot interface using particle filter and Kalman filter for dual robots. *IEEE Trans. Ind. Electron.* **2014**, *62*, 2257–2264. [[CrossRef](#)]
54. Du, G.; Zhang, P.; Liu, X. Markerless human–manipulator interface using leap motion with interval Kalman filter and improved particle filter. *IEEE Trans. Ind. Electron.* **2016**, *12*, 694–704. [[CrossRef](#)]
55. Cui, R.; Chen, L.; Yang, C.; Chen, M. Extended state observer-based integral sliding mode control for an underwater robot with unknown disturbances and uncertain nonlinearities. *IEEE Trans. Ind. Electron.* **2017**, *64*, 6785–6795. [[CrossRef](#)]



© 2020 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 (<http://creativecommons.org/licenses/by/4.0/>).