



# A Framework of Grasp Detection and Operation for Quadruped Robot with a Manipulator

Jiamin Guo <sup>1,2</sup>, Hui Chai <sup>1,2,\*</sup>, Qin Zhang <sup>1,2,3</sup>, Haoning Zhao <sup>1,2</sup>, Meiyi Chen <sup>1,2</sup>, Yueyang Li <sup>3</sup> and Yibin Li <sup>1,2</sup>

<sup>1</sup> Center for Robotics, School of Control Science and Engineering, Shandong University, Jinan 250061, China; 201814461@mail.sdu.edu.cn (J.G.); cse\_zhangq@ujn.edu.cn (Q.Z.); zhaohaoning@mail.sdu.edu.cn (H.Z.); 202334902@mail.sdu.edu.cn (M.C.); liyb@sdu.edu.cn (Y.L.)

<sup>2</sup> Engineering Research Center of Intelligent Unmanned System, Ministry of Education, Jinan 250061, China

<sup>3</sup> School of Electrical Engineering, University of Jinan, Jinan 250022, China; cse\_liyy@ujn.edu.cn

\* Correspondence: chaimax@sdu.edu.cn

**Abstract:** Quadruped robots equipped with manipulators need fast and precise grasping and detection algorithms for the transportation of disaster relief supplies. To address this, we developed a framework for these robots, comprising a Grasp Detection Controller (GDC), a Joint Trajectory Planner (JTP), a Leg Joint Controller (LJC), and a Manipulator Joint Controller (MJC). In the GDC, we proposed a lightweight grasp detection CNN based on DenseBlock called DES-LGCNN, which reduced algorithm complexity while maintaining accuracy by incorporating UP and DOWN modules with DenseBlock. For JTP, we optimized the model based on quadruped robot kinematics to enhance wrist camera visibility in dynamic environments. We integrated the network and model into our homemade robot control system and verified our framework through multiple experiments. First, we evaluated the accuracy of the grasp detection algorithm using the Cornell and Jacquard datasets. On the Jacquard dataset, we achieved a detection accuracy of 92.49% for grasp points within 6 ms. Second, we verified its visibility through simulation. Finally, we conducted dynamic scene experiments which consisted of a dynamic target scenario (DTS), a dynamic base scenario (DBS), and a dynamic target and base scenario (DTBS) using an SDU-150 physical robot. In all three scenarios, the object was successfully grasped. The results demonstrate the effectiveness of our framework in managing dynamic environments throughout task execution.

**Keywords:** quadruped robot; grasp detection; motion planning; deep learning



**Citation:** Guo, J.; Chai, H.; Zhang, Q.; Zhao, H.; Chen, M.; Li, Y.; Li, Y. A Framework of Grasp Detection and Operation for Quadruped Robot with a Manipulator. *Drones* **2024**, *8*, 208. <https://doi.org/10.3390/drones8050208>

Academic Editors: Jiacun Wang, Eugenio Cesario, Guanjun Liu and Jun Wang

Received: 11 April 2024

Revised: 13 May 2024

Accepted: 15 May 2024

Published: 19 May 2024



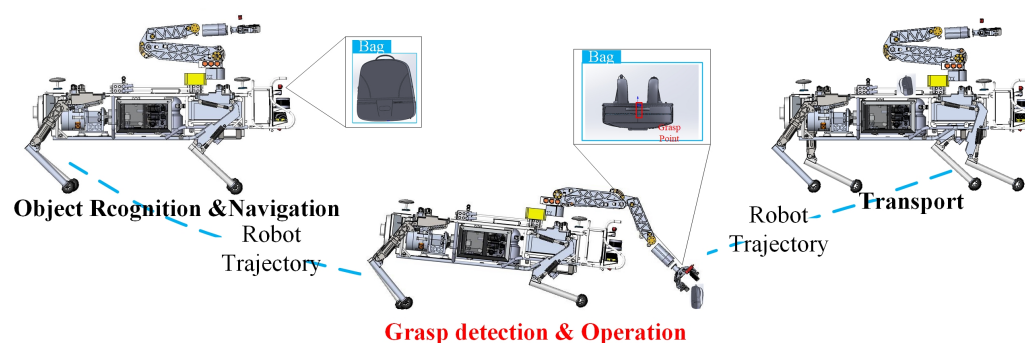
**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Quadruped robots with manipulators combine the motion performance of quadruped robots and the operational capability of manipulators [1–4], providing innovative solutions for the transportation of supplies to rescue efforts in the environment after a disaster [5–8], as shown in Figure 1. In these scenarios, the limited payload capacity and dynamic environment place significant demands on the grasping ability of the mounted manipulator.

When transporting supplies after a disaster, robots need to move over rugged terrain. In order to ensure stability during movement, the robot's own load needs to be minimized, so it cannot be equipped with high-performance industrial computers which are usually large and heavy. In this scenario, the location of the supplies may be outside the operating space of the manipulator, and the robot needs to adjust its base posture to grasp the target. In addition, adverse weather conditions such as aftershocks, strong winds, etc., may also cause the supplies to move. Therefore, the grasping detection algorithm used in these robots must be highly precise, fast, and have low algorithmic complexity. The development of deep learning has made meeting these requirements possible [9]. Researchers have conventionally relied on manually designing geometric features for grasping detection algorithms, resulting in low detection accuracy and difficulty in dealing with unknown

objects [10–13]. However, recent advances in deep learning algorithms have allowed the use of sliding window models and common network structures [14–17] such as ResNet-50 [18] to extract high-quality grasping features, resulting in improved detection accuracy. Despite these improvements, some methods still suffer from resolution loss due to the scene and produce dense predictions depending on the stride of the sliding window, estimated aspect ratio, and angle of the box [10]. To address this issue, pixel-level networks have been developed. Zeng et al. constructed a robotic pick-and-place system based on the ResNet-101 network. The system takes a  $640 \times 480$  RGBD image as input and generates a densely labeled pixel-wise map of the same resolution [19]. However, the large structure of the network limits real-time detection. To address this issue, Morrison et al. developed lightweight grasping detection networks, GG-CNN and GG-CNN2, at the expense of sacrificing detection accuracy [20]. Therefore, it is important to balance detection accuracy while improving the speed of the grasp detection algorithm.



**Figure 1.** Legged robots equipped with manipulators combine the flexibility of legged robots and the operation capability of the manipulator. Due to the challenges posed by mobile bases, complex environments, and varied tasks, accurate detection and manipulation of objects requires sophisticated techniques. This study focuses on solving these problems, as indicated in the red section.

In addition, the aforementioned deep learning-based grasping detection methods typically rely on vision; therefore, it is important to maintain the target object in the FOV to ensure the grasping detection methods have effective inputs [21]. One challenge faced by quadruped robots with manipulators in outdoor environments is the limited field of view (FOV) of the wrist cameras used for grasping detection, which means objects can leave the FOV as the manipulator moves. The ability of a camera to maintain an object in its FOV during the movement of the manipulator is called visibility. Although equipping robots with an unmanned aerial vehicle to extend the FOV is a commonly used solution, it is expensive [22]. To improve the visibility of the target object during the motion of the manipulator, Chen et al. considered time delay factors in trajectory planning and tracking [23–25]. D.-H. Park et al. proposed a method called position-based visual servoing that considers both the visual and physical constraints of the manipulator to compute trajectories that converge to a desired pose [26]. However, this method has not been validated in dynamic environments. T. Shen et al. improved the visibility of a manipulator using manual teaching methods; however, their approach lacks real-time performance and cannot be applied to dynamic scenes [27]. Recent research has explored the use of reinforcement learning to adaptively adjust the motion poses of the manipulator to ensure that the target object remains in view [28]. However, these methods require extensive pretraining and powerful computational resources. To sum up, it is very crucial to develop a path-planning algorithm that does not require additional sensors or devices and considers target visibility while achieving high real-time performance.

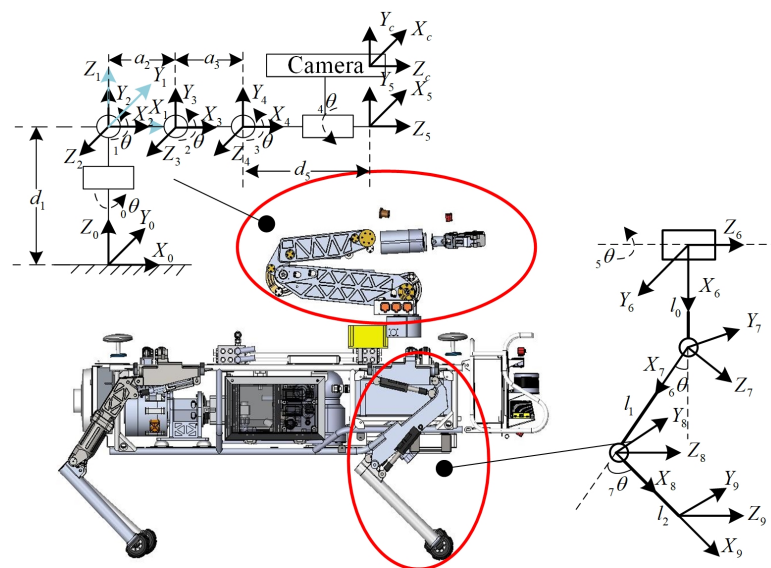
To meet the requirements of grasping and detection algorithms for legged robots equipped with manipulators and to enhance the the wrist camera's visibility of target objects during operations, our contributions are as follows:

- (1) We propose a lightweight grasping convolutional neural network (CNN) based on DenseBlock (DES-LGCNN) to perform pixel-wise detection. Based on two self-made modules (UP and DOWN modules), this algorithm is capable of balancing accuracy and speed during grasp detection.
- (2) We develop a high-visibility motion planning algorithm for manipulators that can ensure the visibility of objects during motion in real time without adding other sensors.
- (3) We integrate the proposed grasping detection algorithm and trajectory optimization model into the control system of our independently developed quadruped robot with a manipulator, which can be used in various environments.

This paper is structured as follows. Some preliminary concepts are described in Section 2. We present the architecture of the framework in Section 3.1 and present the improved methodology in detail in Sections 3.2 and 3.3. The experiments and results are given in Section 4. Finally, we conclude the study in Section 5.

## 2. Preliminaries

The joint angles and link numbers of the quadrupedal robot equipped with a manipulator are defined in Figure 2. Due to the robot's four legs all having the same structure, only the left front leg is labeled in Figure 2. Different numbered coordinate systems are named  $\{O_i\}$ , where  $i$  represents the coordinate system number.  $\{O_c\}$  represents the camera coordinate system.  $d$ ,  $l$ , and  $\alpha$  represent the lengths of the link.



**Figure 2.** D-H model for manipulator and leg of robot. The structure of each leg is identical, and only the front leg is labeled, whereas the indices for the other legs are incremented accordingly.

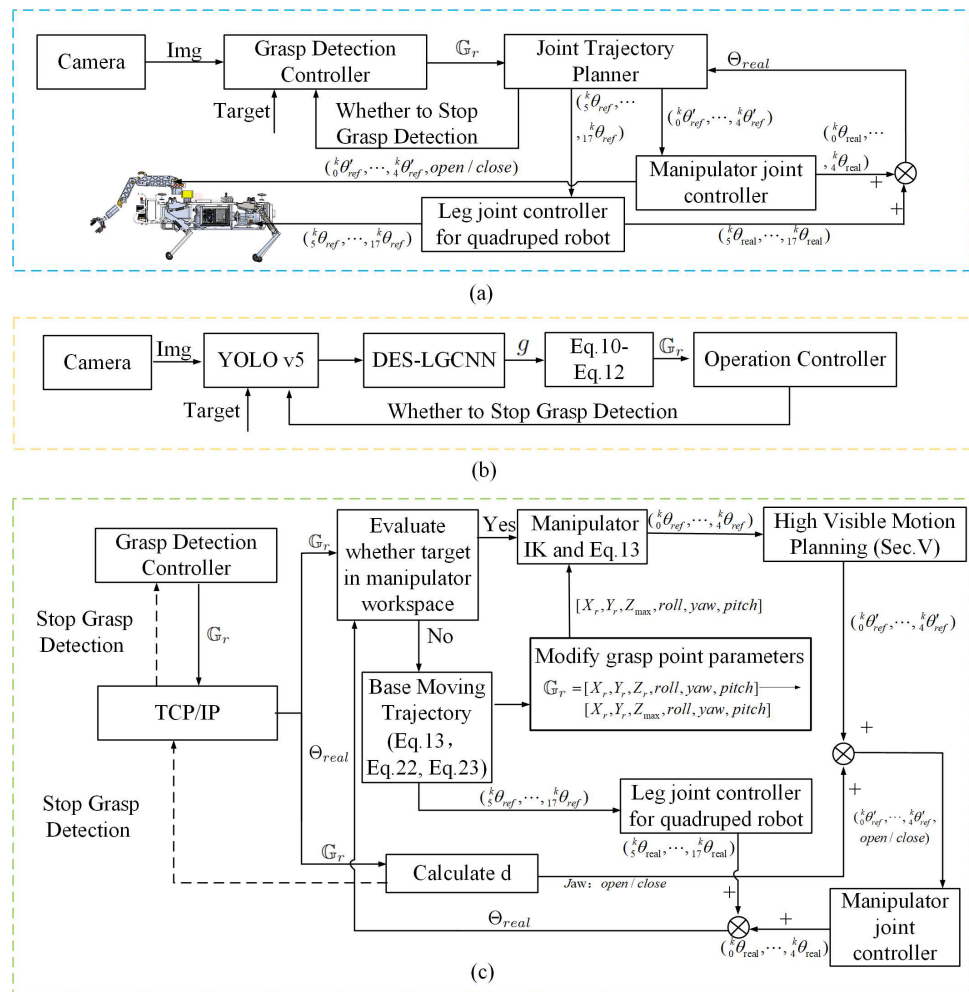
To describe the different joint angles of the robot, we define a symbol  ${}^k\theta_{* \otimes}$ , where  $*$  represents the number of  $\theta$ , ranging from 0 to 17, including the angles of the manipulator, torso, and legs;  $k$  represents time; and  $\otimes$  represents the attribute, such as min, max, or reference.

## 3. Methodology

In order to improve the speed and accuracy of grasping detection and the adaptability to dynamic environments when the quadrupedal robot equipped with a manipulator performs the environmental task of transporting materials after disaster, three aspects of grasp detection and operation framework, grasp detection algorithm and robot motion planning algorithm are studied in this paper.

### 3.1. Architecture of the Framework

The object detection and grasping framework (Figure 3a) consists of three independent controllers: a Grasp Detection Controller (GDC) (Figure 3b), a Joint Trajectory Planner (JTP) (Figure 3c), a Leg Joint Controller for the quadruped robot, and a Manipulator Joint Controller. To achieve precise grasp detection and effective grasping of the target object, we focused on the design of the GDC (Section 3.2) and JTP (Section 3.3). The input of the entire framework is four-dimensional images and the target type. Communication between different controllers is facilitated using the TCP/IP protocol.



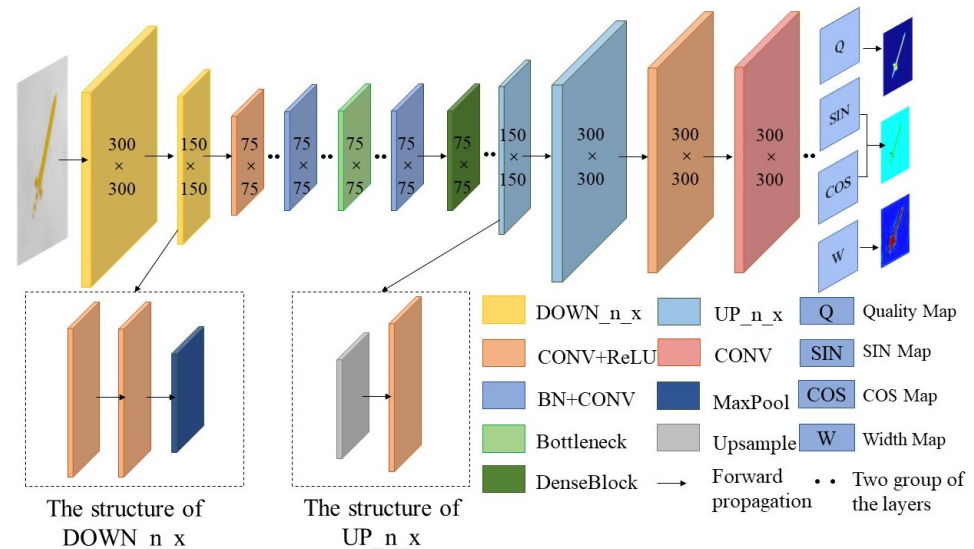
**Figure 3.** Developed framework of grasp detection and operation for a quadruped robot with a manipulator. (a) Entire framework of the system. (b) Framework of the GDC. (c) Framework of the JTP.

The GDC is mainly responsible for grasp detection, as shown in Figure 3b. The YOLO v5 algorithm is applied to obtain the position of the target object in {I}. Subsequently, the image containing the object information is fed into a lightweight grasping CNN based on DenseBlock (DES-LGCNN), which is developed to generate the grasping point  $g$  in {I}. Next,  $g$  is transformed into  $\mathbb{G}_r$  using the pinhole model for further analysis using the OC.  $\mathbb{G}_r$  denotes a six-dimensional representation  $\mathbb{G}_r = [X_r, Y_r, Z_r, roll, yaw, pitch]$ . The details of the DES-LGCNN algorithm and the method used to calculate the posture of the End-Effector (EE) will be explained in Section 3.2. We use the OC to obtain the joint angle of the robot at any given time and send it to the leg joint controller for the quadruped robot and the manipulator joint controller (Figure 3c). In the first step in the JTP, the motion planner receives  $\mathbb{G}_r$  using the TCP/IP communication protocol. Afterward, the trajectory planning task is divided into the trajectory planning task of the manipulator and the trajectory

planning task of the base based on inverse kinematics (IK) and Equation (15). If the target object is not within the manipulator’s workspace, we need to move the base to expand the working space of the manipulator; otherwise, the base remains stationary. Additionally, to ensure that the target object is within the camera’s FOV, we also need to optimize the path using the proposed high-visibility motion planning algorithm (Section 3.3). Based on the above methods, we derive the reference joint angle values for the base and manipulator joint spaces. These joint angle values are sent to the quadruped robot’s leg joint controller and the manipulator’s joint controller, respectively. The two controllers utilize a PID algorithm. The two controllers execute tasks simultaneously and send real joint angles values  $\Theta_{real}$  to the JTP while moving.

### 3.2. Lightweight Grasping Convolutional Neural Network Based on DenseBlock

This study introduces DES-LGCNN, a pixel-level neural network (see Figure 4) designed to accurately detect grasp points by analyzing the probability of each pixel in the input image being a grasp point. The network is inspired by GG-CNN, with the input being an image of the target object and the outputs being a grasp quality map (GQM), a grasp angle cosine map (GACM), a grasp angle sine map (GASM), and a grasp width map (GWM) [20]. Three input image modes are available: depth image (D), color image (RGB), and depth image + color image (RGBD). The output image size is consistent with that of the input image. To reduce the number of network parameters and computational complexity, we replaced the deconvolutional layer with our designed UP module, each consisting of an upsampling layer and a convolutional layer. The specific design ideas are described as follows.



**Figure 4.** Architecture of DES-LGCNN. The left and right dashed rectangles represent the architecture of DOWN\_x\_n and UP\_x\_n modules, respectively. In this figure, different colors represent different network structures.

In conventional pixel-level neural networks, deconvolution is commonly used to restore the size of the feature map obtained after feature extraction to the size of the input image. The upsampling layer has the same ability as the convolutional layer. We define the height  $\times$  width  $\times$  channels of the input for each layer of DES-LGCNN as  $C_{in} \times C_{in} \times W_{in}$ , the output as  $C_{out} \times C_{out} \times W_{out}$ , and the kernel as  $k \times k$ . Without bias, the computational complexity of deconvolution is as follows:

$$C_{out} \times k^2 \times C_{in}^2 \times W_{in} \tag{1}$$

The computational complexity for upsampling using the nearest neighbor algorithm is as follows:

$$C_{out}^2 - C_{in}^2 \quad (2)$$

Let  $C_{out} = C_{in} \times n$ , where  $n$  is increased to the power of two. Equations (1) and (2) are as follows:

$$\begin{aligned} & C_{out} \times k^2 \times C_{in} \times W_{in} - (C_{out}^2 - C_{out}^2/n^2) \\ &= \frac{C_{out}^2}{n^2} (k^2 \times H_{in} \times W_{in} \times n - n^2 + 1) \end{aligned} \quad (3)$$

Typically,  $k^2 \times H_{in} \times W_{in} \times n - n^2 \gg 1$ , making the use of an upsampling layer an effective way to reduce network computation. However, upsampling can result in the loss of information between intervals. We assume that the information about each pixel is related to the surrounding pixels. We added a convolutional layer after the upsampling layer to supplement the missing information. The value of each pixel  $p_c$  in the new feature map generated by the convolutional layer can be calculated as follows:

$$p_c(\mu_i, v_j) = \sum_{\mu=1}^k \sum_{v=1}^k w_{\mu,v} p_u(\mu_i + \mu - 1, v_j + v - 1) \quad (4)$$

where  $(\mu_i, v_j)$  represents the position of the pixel in the feature map, and  $p_u$  denotes the pixel of the feature map generated by the upsampling layer.  $w_{\mu,v}$  denotes the weight of the kernel at position  $(\mu, v)$ . This reflects the relationship between the current pixel of the new feature map and the surrounding pixels of the pixel in the same position as the old feature map, which can be used to reconstruct lost information. The computation of the bias-free convolutional layer is as follows:

$$C_{out}^2 \times k^2 \times C_{in} \times W_{out} \quad (5)$$

Equation (1)/Equation (5) are calculated as follows:

$$\frac{C_{in} \times W_{in}}{C_{out} \times W_{out}} \quad (6)$$

The design principle of Equation (6)  $< 1$  can effectively reduce computation. Based on this principle, we designed the Up\_n\_x layer shown in Figure 4, where  $x$  represents the number of channels after the feature map passes through the module, and  $n$  indicates that the output feature map is  $1/n \times 1/n$  of the original. To correspond with the UP module, we designed the DOWN module, which consists of two convolutional layers and a max pooling layer to reduce the feature map size and computation. This module is located in the feature extraction part of the network, specifically the yellow part in front of the Down\_n\_x network. Here,  $n$  indicates that the output feature map is  $1/n \times 1/n$  of the original. In addition, to avoid the gradient explosion problem that may occur during the forward propagation of the network, we included dense modules in the middle of the network, as shown in the dark green part of Figure 4.

After evaluating different loss functions, we chose the mean squared error (MSE) as the final loss function [20]:

$$\operatorname{argmin} \operatorname{MSE}(G_i, \hat{G}_i) \quad (7)$$

where  $G_i$  denotes the network-generated grasp feature maps and  $\hat{G}_i$  denotes the ground truth in the  $i$ -th training.

After passing through the DES-LGCNN network, we need to convert the output GQM, GASM, GACM, and GWM into values that can be parsed using the OC. The position of the

grasp point  $(\mu_g, v_g)$  and the grasping angle  $\theta_g$  of the grasp point in the image coordinate system are calculated as follows:

$$(\mu_g, v_g) = \text{Find}(\max(QM(\mu, v))) \quad (8)$$

$$\theta_g = \frac{1}{2} \arctan\left(\frac{\sin(2ASM(\mu_g, v_g))}{\sin(2ACM(\mu_g, v_g))}\right) \quad (9)$$

On the basis of Equations (8) and (9), the grasping representation in the image coordinate system can be obtained. We use the five-dimensional model developed by [14] to represent a grasp point  $g$  as follows:

$$g = \{\mu_g, v_g, \theta_g, h, w\} \quad (10)$$

where  $h$  and  $w$  represent the height and width of the jaw, respectively. After  $g$  is obtained, the pinhole model is used to transform it from the image coordinate system to obtain the position of  $p_g = (p_{gx}, p_{gy}, p_{gz})$  in  $\{O_c\}$ :

$$p_{gz} \begin{bmatrix} \mu_g \\ v_g \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & \mu_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{gx} \\ p_{gy} \\ p_{gz} \end{bmatrix} \quad (11)$$

where  $f_x$ ,  $f_y$ ,  $\mu_0$ , and  $v_0$  represent the camera parameters. In addition, to achieve the grasping task, the position and orientation of the EE should be considered. Let  $\mathbb{G}r = [X_r, Y_r, Z_r, \theta_{gr}, \theta_{gy}, \theta_{gp}, W]$  define a grasp in manipulator base coordinates systems; the grasp is determined by the EE's pose, such as the position of the grasp  $(X_r, Y_r, Z_r)$ , the roll of the gripper  $\theta_{gr}$ , the yaw of the gripper  $\theta_{gy}$ , the pitch of the gripper  $\theta_{gp}$ , and the gripper width  $W$ . Since the gripper we used only has two statuses (open and width), the width is equal to the maximum opening width of the gripper. We can apply the following transformation to convert the grasping representation in  $\{O_c\}$  to  $\{O_0\}$ :

$$[X_r, Y_r, Z_r]^T = {}^0T_5 \left( {}^5T_c p_g \right) \quad (12)$$

where  ${}^5T_c$  denotes the transformation matrix from  $\{O_c\}$  into  $\{O_5\}$ .  ${}^0T_5$  denotes the transformation matrix from  $\{O_5\}$  into  $\{O_0\}$ .  $[X_r, Y_r, Z_r]$  represents the position of the grasp point in  $\{O_0\}$ .  $\theta_{gr}$  is equal to  $\theta_g$ .  $\theta_{gp}$  can be set to a fixed value.  $\theta_{gy}$  can be calculated as

$$\theta_{gy} = \arctan2(Y_r, X_r) \quad (13)$$

The target state of the manipulator is defined as  $\Theta_{goal} = ({}^0\theta_g, {}^1\theta_g, \dots, {}^4\theta_g)$ . Based on IK, it can be calculated as

$$\Theta_{goal} = IK(\mathbb{G}r) \quad (14)$$

### 3.3. High-Visibility Motion Planning Module

High-visibility motion planning refers to the capability of a mobile manipulator to strategically adjust and broaden its FOV by manipulating the pose of its manipulator to ensure that an object is within the FOV. The algorithm of the high-visibility motion planning module is shown in Algorithm 1.

Because quadruped robots are capable of movement, we only need to consider cases in which the  $\mathbb{G}r$  is outside the manipulator's workspace in the Z direction. In steps 1–3, when the target object is located within the dexterity space of the manipulator, the target state  $({}^0\theta_g, {}^1\theta_g, \dots, {}^4\theta_g)$  of the manipulator can be obtained through inverse kinematics. The current state  $({}^0\theta_{real}, {}^1\theta_{real}, \dots, {}^4\theta_{real})$  of the manipulator can be obtained using the encoder. On the basis of linear interpolation, the motion path of the manipulator can be

obtained. Assuming that the interpolated motion path of the manipulator has  $\alpha$  points and the time set for each point is denoted as  $K$ , for  $k \in [0, n]$ , we obtain

$${}^k\theta_{ref} = (i\theta_g - i\theta_{real})/\alpha \times k + i\theta_g \quad (15)$$

Based on Equation (15), the motion trajectory of the manipulator can be obtained as  $\zeta = \left\{ \left[ {}^0\theta_{ref}, \dots, {}^4\theta_{ref} \right], \dots, \left[ {}^n\theta_{ref}, \dots, {}^4\theta_{ref} \right] \right\}$ . When the manipulator travels approximately one-tenth of its trajectory, it undergoes a replanning process. The number of waypoints is fixed each time the replanning process is performed. As the manipulator approaches the target, the angular velocity of each joint decreases to protect both the manipulator and the target object.

---

#### Algorithm 1 High-visibility motion planning algorithm

---

```

1:  $\Theta_{goal} \leftarrow IK(\mathbb{G}_r)$ 
2:  $\zeta \leftarrow \text{Equation (15)}$ 
3: Calculate the distance between the object and the camera  $d \leftarrow d(\Theta_{goal}, (x_c, y_c, z_c))$ 
4: while ( $k = 0 \dots n$ ) do
5:   if  $d \geq d_{min}$  then
6:      $\theta \leftarrow \text{Equation (16)}$ 
7:     if  $\theta \geq \text{FOV}$  then
8:        $[{}^k\theta'_{ref}, \dots, {}^k\theta_{ref}]' \leftarrow \text{Equation (22)}$ 
9:       return  $[{}^k\theta'_{ref}, \dots, {}^k\theta_{ref}]'$ 
10:    else
11:      return  $[{}^k\theta_{ref}, \dots, {}^k\theta_{ref}]$ 
12:    end if
13:  else
14:    if  $k = n$  then
15:      close jaw
16:      break
17:    else
18:      return  $[{}^k\theta_{ref}, \dots, {}^k\theta_{ref}]$ 
19:    end if
20:  end if
21: end while

```

---

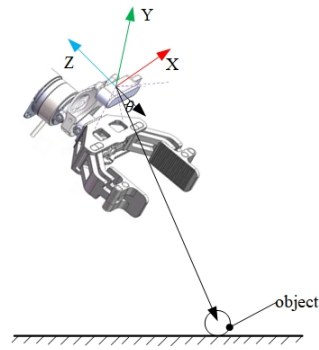
In steps 5 and 6, we use a threshold value  $d_{min}$  to determine whether the EE is close to the object. If the distance between the target object and the EE is too close, only a part of the object will be visible in the FOV, and effective information cannot be extracted. In this case, we proceed directly to steps 14–19. In this section, the manipulator moves according to  $\zeta$  and closes the jaw when the target position is reached. However, if  $d \geq d_{min}$ , we use steps 6–12 to determine whether the current trajectory point  $[{}^k\theta_{ref}, \dots, {}^k\theta_{ref}]$  should be optimized.

The angle between the vector pointing from the camera's position to the target object's center of mass  $\vec{n}_{zc}$  and the line of sight  $\vec{n}_{co}$  is defined as  $\theta$ . In  $\{O_c\}$ , this angle can be calculated as follows:

$$\theta = \arccos\left(\frac{\vec{n}_{zc}(k) \cdot \vec{n}_{co}(k)}{|\vec{n}_{zc}(k)| |\vec{n}_{co}(k)|}\right) \quad (16)$$

where  $\vec{n}_{zc}$  denotes the unit vector  $(0, 0, -1)$ , which is in the opposite direction of the Z-axis of the wrist camera coordinate system.  $\vec{n}_{co} = G_r - P_c$ , where  $P_c = (x_c(t), y_c(t), z_c(t))$  represents the position of the camera in  $\{O_5\}$ . A schematic for solving  $\theta$  is shown in Figure 5.





**Figure 5.** Schematic for solving  $\theta$ .

If the target is not within the FOV, the trajectory must be adjusted (steps 7–9). Otherwise, the trajectory point in  $\zeta$  is executed directly (steps 10–12). To ensure that the target is as close to the center of the FOV as possible, it is necessary to satisfy

$$\begin{aligned} \min J(\vec{n}_{zc}(k), \vec{n}_{co}(k)) &= 1 - \frac{\vec{n}_{zc}(k) \cdot \vec{n}_{co}(k)}{|\vec{n}_{zc}(k)| |\vec{n}_{co}(k)|} \\ \text{s.t. } k &\geq 0 \end{aligned} \quad (17)$$

Typically, to prevent the manipulator from moving back and forth because of the adjustment of the camera view, we add the following constraint:

$$z_c \leq z_c(t) \quad (18)$$

Based on IK:

$$\begin{aligned} x_c &= a_2 \times \cos_1 \theta \cos_2 \theta + a_3 \cos_1 \theta \cos(2\theta + 3\theta) \\ &\quad + d_5 \cos(1\theta) \cos(2\theta + 3\theta + 4\theta) \end{aligned} \quad (19)$$

$$\begin{aligned} y_c &= a_2 \times \sin_1 \theta \cos_2 \theta + a_3 \sin_1 \theta \cos(2\theta + 3\theta) \\ &\quad + d_5 \sin(1\theta) \cos(2\theta + 3\theta + 4\theta) \end{aligned} \quad (20)$$

$$\begin{aligned} z_c &= a_2 \times \sin_2 \theta + d_1 + a_3 \times \sin(2\theta + 3\theta) \\ &\quad + d_5 \times \sin(2\theta + 3\theta + 4\theta) \end{aligned} \quad (21)$$

Add Equation (18) to Equation (21) into Equation (17):

$$\begin{aligned} \min J(1\theta(k), 2\theta(k), 3\theta(k), 4\theta(k)) \\ \text{s.t. } \begin{cases} A\Theta' - b_{\min} \geq 0 \\ -A\Theta' + b_{\max} \geq 0 \\ a_2 \times \sin_2 \theta + d_1 + a_3 \times \sin(2\theta + 3\theta) \\ \quad + d_5 \times \sin(2\theta + 3\theta + 4\theta) \leq z_c(t) \\ k \geq 0 \end{cases} \end{aligned} \quad (22)$$

In Equation (22),  $A$  represents the identity matrix.  $\Theta'$ ,  $b_{\min}$  and  $b_{\max}$  are

$$\begin{aligned} \Theta' &= [1\theta, 2\theta, 3\theta, 4\theta]^T \\ b_{\min} &= [1\theta_{\min}, 2\theta_{\min}, 3\theta_{\min}, 4\theta_{\min}]^T \\ b_{\max} &= [1\theta_{\max}, 2\theta_{\max}, 3\theta_{\max}, 4\theta_{\max}]^T \end{aligned} \quad (23)$$

In addition, because  $-1 \leq \frac{\vec{n}_{zc}(k) \cdot \vec{n}_{co}(k)}{|\vec{n}_{zc}(k)| |\vec{n}_{co}(k)|} \leq 1$ , it follows that  $J \in [0, 2]$ . Therefore,  $J$  must have a minimum value, which can be solved using the Sequential Least Squares Programming (SLSQP) algorithm.

The main idea of calculating the base trajectory is to move the base in such a way that the grasping point is positioned at a specific position in manipulator's base coordinate system where the object is within the manipulator's dexterous workspace. When the target is outside the workspace of the manipulator, it is necessary to adjust the distance between the robot's hip joint and the ground. To reduce the number of variables required to solve the optimization algorithm, the base movement does not expand the FOV of the camera. Take the left front leg as an example. Assuming that the robot's foot coordinate system is flush with the ground in the vertical direction, we define the robot's left hip joint in the foot coordinate system as  $(X_{lt}(t), Y_{lt}(t), Z_{lt}(t))$ . In the vertical direction, the edge of the dexterous workspace of the manipulator is defined as  $Z_{max}$ .  $\Delta Z_{lt}(t) = Z_{max} - Z_r$ , with the reference value  $Z_{ref} = Z_{lt}(t) - \Delta Z_{lt}(t)$ . According to forward kinematics (FK), the desired angle values for the left front leg are

$${}_6\theta_{ref} = \arccos\left(\frac{l_1^2 + (l_0 - X_{ref})^2 + X_{lt}(t)^2 - l_2^2}{2l_1\sqrt{(l_0 - Z_{ref})^2 + x^2}}\right) - \arctan\left(\frac{X_{lt}(t)}{-l_0 + X_{ref}}\right) \quad (24)$$

$${}_7\theta_{ref} = \arccos\left(\frac{l_1^2 + l_2^2 - (l_0 - X_{ref})^2 - X_{lt}(t)^2}{2l_1l_2}\right) - \pi \quad (25)$$

The real angles  $\{{}_6\theta_{real}, {}_7\theta_{real}\}$  of the legs are read using the encoder, and the method of solving the trajectory points of the robot's leg motion is the same as that in Equation (15).

## 4. Experiments

### 4.1. Experimental Environment

#### 4.1.1. Network Training and Testing

The personal computer LEGION Y7000 (Lenovo, Beijing, China) is used to train and test the DES-LGCNN algorithm. The hardware system of this personal computer includes an Intel Core I7-10875H CPU (Intel®, Santa Clara, CA, USA), 32 GB RAM, and an NVIDIA GeForce RTX 2060 graphics card (NVIDIA, Santa Clara, CA, USA). The software system mainly includes Ubuntu 18.04, NVIDIA 472.19 graphics card driver, CUDA 11.0, CUDNN V8.0.5, and PyTorch 1.10.0.

#### 4.1.2. Simulation Environment

The simulation platform is constructed on the basis of Webots 2021a. As shown in the left part of Figure 6, we built a five-degree-of-freedom manipulator in the simulation environment. The gripper has two fingers, with the maximum opening and closing reaching 300 mm. In particular, a depth camera for grasping detection is installed on the EE and is placed 15 cm away from the X-axis of the EE. This will ensure that the vision of the camera effectively covers the workspace of the manipulator and avoids its occlusion.

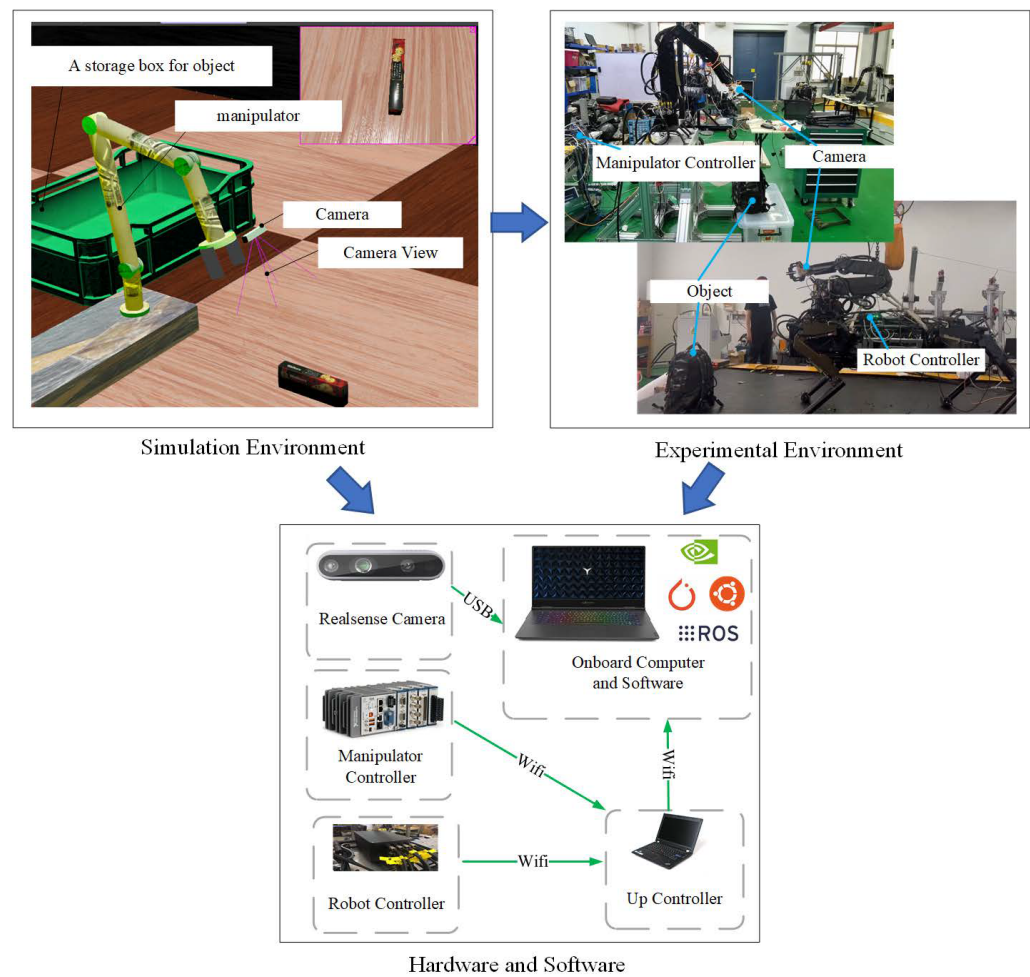
#### 4.1.3. Physical Prototype

The parameters of the self-made hydraulic manipulator are consistent with those of the simulation environment. The object recognition and grasp detection modules are also deployed in LEGION Y7000 with the same hardware as that described in Section 4.1.1. The Compact-RIO 9039 controller (NI, Austin, TX, USA) is used to control the motion of the manipulator. A local area network is constructed between LEGION Y7000 and the Compact-RIO 9039 controller, and the TCP/IP protocol is used for communication. The experimental environment of the manipulator is shown on the right side of Figure 6.

To evaluate the performance of our algorithm on a quadruped robot, we used the SDU-150 quadruped robot as the chassis and equipped it with our own manipulator. The parameters of the robot is shown Table 1. The robot's base comprises 12 hydraulic servo valves, 12 force sensors, and 12 displacement sensors. Among these, the hydraulic

servo valves control the movement of the robot’s legs, the displacement sensors obtain actual motion parameters of each joint in the legs, and the force sensors facilitate force control and feedback for each joint. The manipulator consists of one force sensor (mounted at the EE), five hydraulic servo valves, and five encoders. The hydraulic servo valves are responsible for controlling the manipulator’s motion. Force sensors are employed for force control and feedback regarding the manipulator’s load. Encoders are utilized to read the actual joint angles, and via inverse kinematics, the EE position can be determined. All the aforementioned data are output through the SDU-150’s logging module at a frequency of 100 Hz.

The robot’s controller is a self-made servo controller with high shock resistance. The robot and manipulator share the same upper computer, a LEGION Thinkpad, with the manipulator controller. The quadruped robot equipped with a manipulator is shown in the middle of Figure 6. The upper computer communicates with LEGION Y7000 through the TCP/IP protocol. The hardware and software are described in the bottom half of Figure 6.



**Figure 6.** Setup for simulation and real-world experiment. This figure describes the simulation environment, the experimental environment in the real world for our five-degree-of-freedom hydraulic manipulator, the SDU-150 quadruped robot equipped with a manipulator, and the hardware and software of the control system.

**Table 1.** The parameters of SDU-150 equipped with a manipulator.

Parameters	Value
length	1420 mm
width	693 mm
height	700 mm
weight	240 kg
climbing slope	25°
obstacle crossing	250 mm
max speed	1.8 m/s
manipulator load	30 kg

#### 4.2. Validation of DES-LGCNN

The Cornell and Jacquard datasets are commonly used to evaluate grasp detection algorithms. We selected these two datasets for comparative experiments and used accuracy and speed as evaluation indices. For a fair comparison with other studies, when the intersection-over-union score is greater than 25% and the angle error is less than 30°, the grasp is considered correct.

During the training process, we use 80% of each dataset as a training set and the remaining 20% as a test set. The number of epochs is set to 400. The learning rate is set to a gradient descent with step decay. The initial learning rate is 0.01 and decreases by 1/10 every 100 epochs. During the training process, we recorded the variations in the loss function under different conditions, primarily involving the two datasets with inputs of D, RGB, and RGBD images. The datasets were classified using two methods, image-wise split (IW) and object-wise split (OW), resulting in six scenarios, as shown in Figure 7. As shown in the figure, in the aforementioned six scenarios, the loss function converges rapidly. After approximately 20 batches, the loss functions in all scenarios converge to values close to 0.

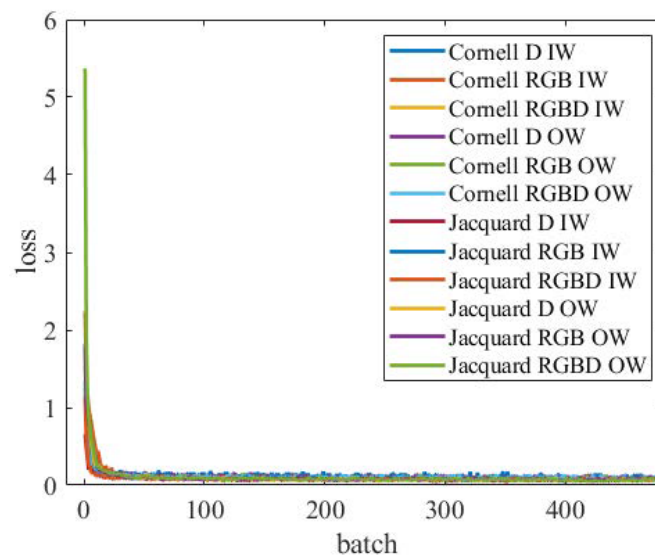
**Figure 7.** Change in loss function during training.

Figure 8 shows part of our experimental results. The results show that our network can effectively detect grasping points. A comparison of the experimental results is shown in Tables 2 and 3. Table 2 shows the experimental results for the Cornell dataset in terms of grasping accuracy, network size, and calculated speeds of different algorithms. In Table 2, when the input is RGBD images, our network has the highest accuracy and the fastest detection speed. The accuracy is 92.49% for IW and 92.39% for OW, and the detection speed is 6 ms. When RGB images are used for recognition, the IW and OW accuracies of GraspNet

are slightly higher than those of our network, but the detection speed is slower. Our network has lower accuracy when only D information is used. In addition, the classification method of the dataset has little influence on accuracy.

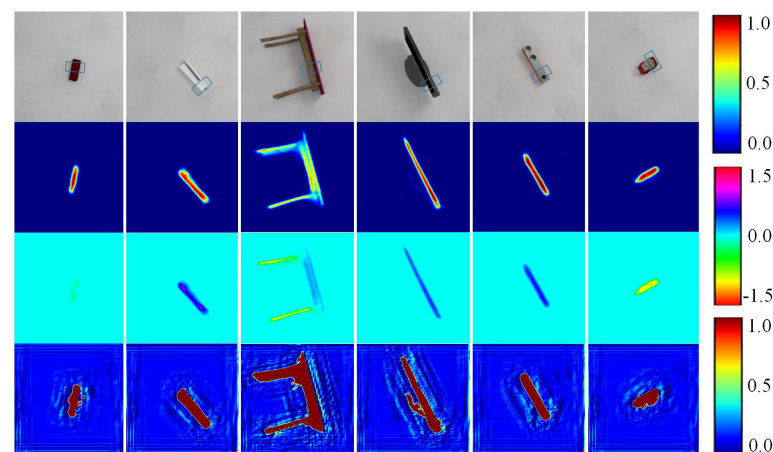
**Table 2.** Comparison of grasp dataset parameters on Cornell dataset.

Approach	IW (%)	OW (%)	Speed (ms)
SAE [14]	73.9	75.66	1350
Alxnet, MultiGrasp [29]	88.0	87.1	76
Two-stage closed-loop [30]	85.3	-	140
GG-CNN [20]	73.0	69.0	19
GraspNet [31]	90.2	90.6	24
ResNet-50x2 [32]	89.2	88.9	103
ours-RGB	90.59	89.89	6
ours-D	80.48	81.26	5.5
ours-RGBD	92.49	92.39	6

**Table 3.** Comparison of grasp dataset parameters on Jacquard dataset.

Approach	IW (%)	OW (%)	Speed (ms)
FCGN(ResNet-50) [17]	89.83	89.26	28
GG-CNN2 [20]	84	83	19
New DNN [33]	85.74	-	-
ours-RGB	86	88.64	6
ours-D	91.39	91.08	5.5
ours-RGBD	92.22	92.35	6

Table 3 shows the experimental results for the Jacquard dataset in terms of the accuracy of the different algorithms. Our algorithm achieves the highest accuracy of 92.22% when the input is RGBD images. In contrast to the performance on the Cornell dataset, when the network uses only D information, the accuracy is not significantly lower than when using RGBD information. It is possible that the Jacquard dataset has a larger scale, which can provide the network with more depth information. However, when only RGB information is used, our algorithm performs moderately well.



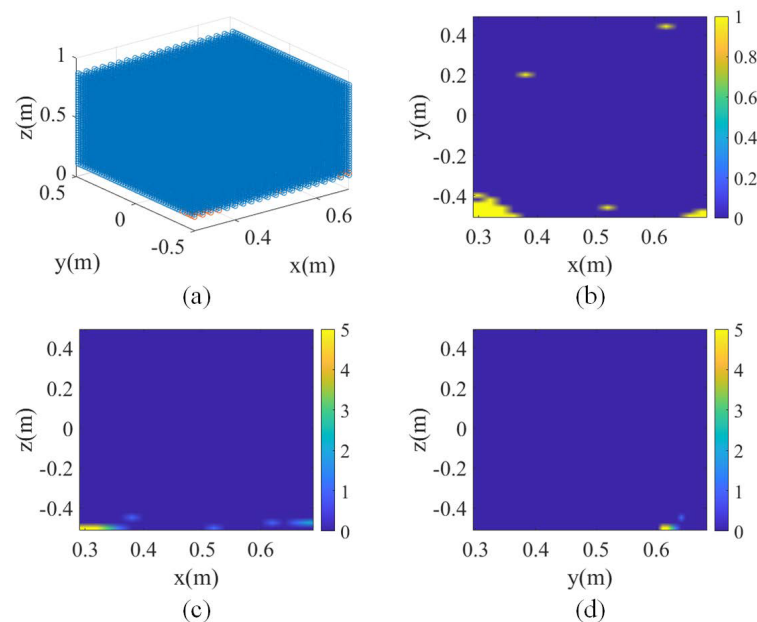
**Figure 8.** Detection result of DES-LGCNN. Columns 1 to 4 represent grasping representation, grasp quality, grasp angle, and grasp width images, respectively. The three bars on the right from top to bottom correspond to the color of the grasp quality feature map, grasp angle feature map, and gripper width feature map, respectively.

Tables 2 and 3 show that the proposed algorithm has high grasp detection accuracy and fast speed, effectively ensuring the real-time performance of grasping detection and providing a basis for the implementation of closed-loop control.

### 4.3. Simulation Experiment

We optimized our algorithm in the dexterous workspace of the manipulator using the SLSQP algorithm. The results are presented in Figure 9. In this space, we select a point every 2 cm for computation using Equation (22). The FOV of the camera is set to  $67^\circ$ , which is similar to that of the commonly used Realsense D435i camera.

In Figure 9, 39,000 points are sampled within the aforementioned space. Of these, 38,980 points are marked in blue, and only 20 points are marked in red. In Figure 9c,d, note that the red points are primarily at the edges of the dexterous workspace. This suggests that our model may require further refinement to handle situations at the edges of the workspace. In addition, most of the unsolvable cases occur when the gripper is approximately 12 cm away from the target object and the camera is positioned very close to the target object with a limited FOV. In such cases, it is understandable that obtaining an effective solution is difficult because our algorithm does not allow the gripper to move far away from the object in the vertical direction. Overall, these results are promising and indicate that our model is well-suited for various tasks in the dexterous workspace of the manipulator.



**Figure 9.** Visibility of optimization algorithms in the dexterous space of the manipulator. (a) The X-axis and Y-axis in this figure represent the position of the target object in the base coordinate system of the manipulator, and the Z-axis represents the vertical distance between the EE and the target object. The optimal solution obtained using our model within the FOV of the camera is shown in blue and outside it in red. (b–d) The projections of (a) onto different planes. Color intensity indicates the number of unsolvable points at each position.

To evaluate the real-time performance of our motion control algorithm, we conducted a visibility detection experiment. To verify the visibility of our proposed motion planning algorithm, we compared it with the A\*, RRT, and RRT\* algorithms in a simulation environment.

We propose a ratio  $\vartheta$  to measure the ability of the motion planning algorithm to keep an object in the FOV:

$$\vartheta = \frac{t_{in}}{t} \quad (26)$$

where  $t$  represents the total time spent from the first detection to the time the EE grasped the object successfully and  $t_{in}$  represents the time the object was in the FOV during movement. The larger the  $\vartheta$ , the better the visibility of our motion planning algorithm. Referring to

the FOV of commonly used cameras, the FOV of the wrist camera is set to  $150^\circ$ ,  $120^\circ$ ,  $90^\circ$ , and  $60^\circ$ . Because the trajectory generated by the genetic algorithm is random, we randomly placed the target object in 20 different positions and grasped the object five times in each position using different algorithms when the FOV was fixed. The evaluation indicator is calculated by averaging all  $\theta$  under the same FOV using the same algorithm, as presented in Table 4.

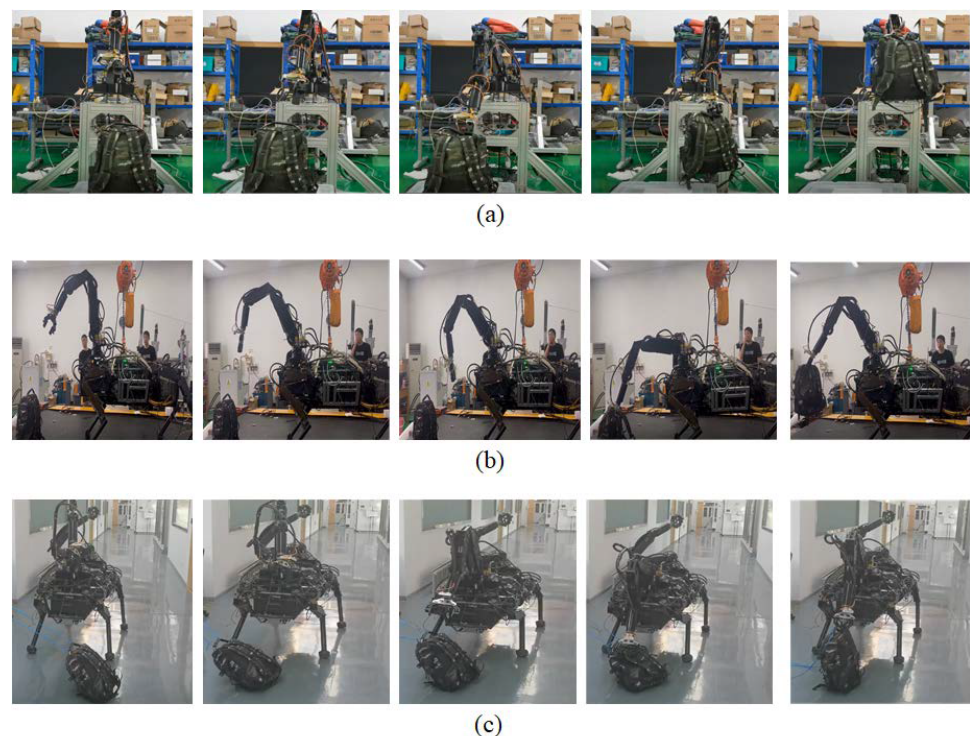
**Table 4.** Visibility under different algorithms.

Algorithm	$150^\circ$	$120^\circ$	$90^\circ$	$60^\circ$
A*	100%	100%	70.83%	62.65%
RRT	100%	100%	65.07%	53.14%
RRT*	100%	100%	59.61%	51.71%
Ours	100%	100%	97.15%	83.33%

When the FOV is set to  $150^\circ$  and  $120^\circ$ , it is similar to that of a global camera. At this point, all motion planning algorithms can effectively ensure that the detected object is always in the FOV during movement. However, when the FOV is reduced to  $90^\circ$ , the  $\theta$  of the other three algorithms decreases sharply. When the FOV is further reduced, our algorithm loses sight of objects approximately 16.66% of the time; these instances are concentrated at the end of the grasping process. At this time, the object is very close to the gripper, and when the FOV of the wrist camera is small, the gripper is in a blind area of the camera's vision.

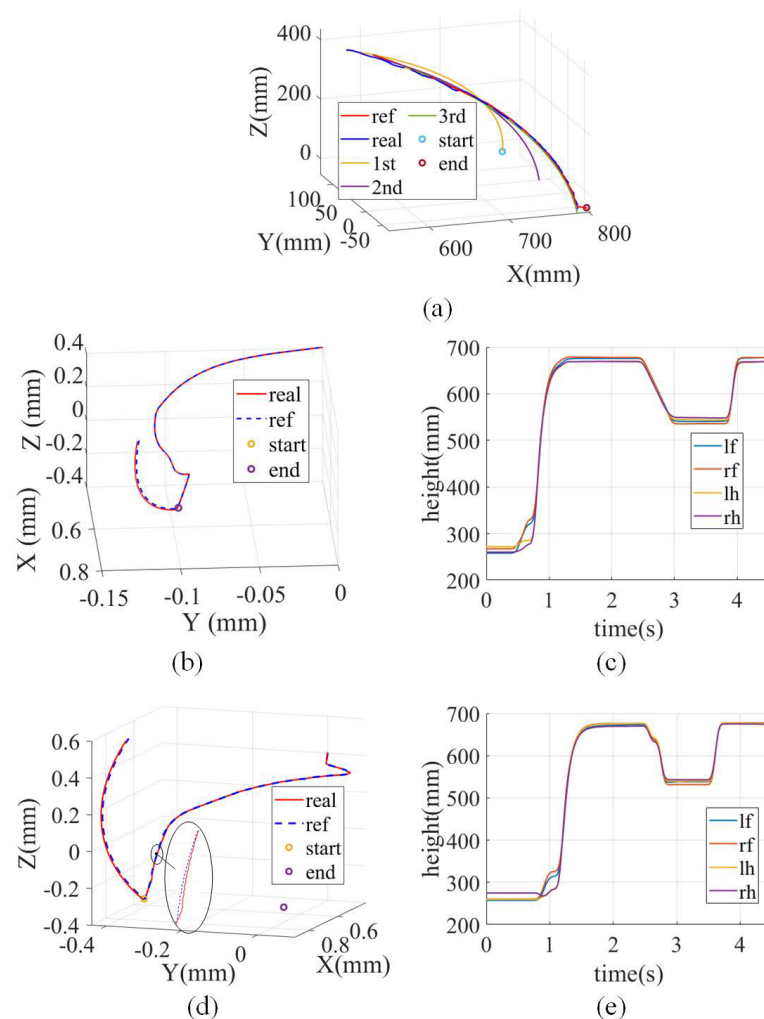
#### 4.4. Dynamic Grasping in Real-World Environments

During post-disaster rescue operations, supplies are often packed in camouflage bags. Therefore, this study conducted experiments on camouflage bags. The dynamic grasping experiment is conducted in three scenarios: a dynamic target scenario (DTS), a dynamic base scenario (DBS), and a dynamic target and base scenario (DTBS). Figure 10 shows the experimental scenarios and the relevant data for DTS, DBS, and DTBS.



**Figure 10.** Three experiment scenarios: (a) DTS, (b) DBS, and (c) DTBS.

To verify the grasping ability of the algorithm in the DTS, an object is placed on a sorting box with wheels so that the object can be moved by dragging the box. The manipulator is placed on a stable platform to ensure that the base of the manipulator does not move. The entire grasping process is depicted in Figure 10a. The object moves within the workspace of the manipulator, and the manipulator tracks the object and grasps it successfully. During the process of grasping, the manipulator is repositioned several times. We selected three representative instances to demonstrate this (Figure 11a). The red line indicates the reference trajectory of the EE, and the blue line indicates the real trajectory. The yellow, purple, and green lines indicate the first, second, and third planning trajectories, respectively. Each planned trajectory effectively reflects the motion of the object. When the object moves, the newly planned trajectory will cover the unreachable part of the original trajectory and finally form the completed reference trajectory. To ensure that there is sufficient effective information in the FOV, when the distance between the EE and the object is less than 20 cm, the grasp detection algorithm stops its process and the planned trajectory does not change. In the early grasping stage, the swing amplitude of the manipulator is significant because the given waypoints are sparse. In the late grasping stage, the reference waypoints are dense; thus, the swing amplitude of the manipulator is effectively reduced. Finally, the manipulator successfully grasps the object. This experiment proves the effectiveness of our algorithm applied to a hydraulic manipulator in a real-world environment.



**Figure 11.** Relevant data on grasping in dynamic scenarios. Each line indicates information about the EE and hip joints in a different scenario. The circle represents the start and end of the object's movements. (a) Trajectory of EE in DTS. (b) Trajectory of EE in DBS. (c) Trajectory of hips in DBS. (d) Trajectory of EE in DTBS. (e) Trajectory of hips in DTBS.



To evaluate the performance of the algorithm in the DBS, we used a quadruped robot with a manipulator. An object is placed on the ground in front of the robot. Because the workspace of the manipulator is limited, the robot must perform a squatting motion to reach the object. The entire experiment process is depicted in Figure 10b. In this experiment, when the object is beyond the manipulator's workspace, the mobile base moves to the ground at a speed  $v_h = h/t$  until the target enters the manipulator's workspace. Here,  $h$  represents the height at which the robot base should descend.  $t$  represents the execution time at which the EE moves to the target position. In this process, the manipulator continues to move along the planned trajectory and maintains grasping detection.

Figure 11b indicates the trajectory of the EE. The red curve indicates the real trajectory, and the blue curve indicates the reference trajectory. The second line graph shows the distance between the hip joints and the ground when the robot squats. The blue, yellow, green, and red curves correspond to the lf, rf, lh, and rh of the robot, respectively. The data represent the values of the robot's hip coordinate system in the upper and lower directions of the ground coordinate system. The robot is powered on from the squatting mode to the initial mode in 0–2.5 s, as shown in Figure 11c. During this period, the four curves tended to first rise and then remain stable. Notably, the robot body starts to descend at 2.5 s and quickly returns to a standing state at 4 s. During the entire grasping process, the object is always in the FOV, and the manipulator finally grasps the object successfully.

Figure 10c illustrates the grasping process in the DTBS. The motion curve of the EE during this period is shown in Figure 11d. As shown in the figure, the EE has a tracking trend for the movement of the object. In the beginning, to enable the EE to move to the location of the object, its trend moves to the right. As the object moves to the left, the EE follows it and moves toward the left. Similar to the experiment shown in Figure 11a, when the distance between the EE and the object is less than 20 cm, the manipulator stops detecting the location of the object. The EE moves toward the object directly according to the planned trajectory without adjusting the initial planned trajectory according to the visibility. Figure 11e shows the distances between the four hips and the ground when the robot squats. Unlike in the DBS, the hip velocities change because of the movement of the object.

## 5. Conclusions

In our research endeavor, we introduce a pioneering closed-loop grasping strategy tailored for quadruped robots outfitted with manipulators. Central to our approach is the deployment of the DES-LGCNN, a lightweight neural network adept at discerning high-precision grasps while mitigating computational overhead. Impressively, this network boasts a detection time of a mere 6 milliseconds, sustaining an accuracy rate surpassing 92%. Moreover, our innovation extends beyond mere detection; we have devised a trajectory planning algorithm meticulously engineered to ensure sustained visibility of the target object within the camera's field of view (FOV) throughout manipulator motion. In the experiment, our algorithm reaches the visibility of 83.33% when the FOV of the camera is only  $60^\circ$ , while other algorithms only reach a visibility of 62.65%. This continuous visual feedback loop enriches the control algorithm, empowering it to adapt dynamically to changing scenarios, even in instances where the robot's torso exhibits non-static behavior.

Notably, the effectiveness of the algorithm is only tested when the robot's torso moves up and down in our experiments, because the robot needs to have a high-precision positioning ability to realize forward and backward movements. This problem requires the robot to have a strong location ability. In the future, our objective will revolve around seamlessly integrating the manipulator subsystem within the broader robotic framework. Such integration would pave the way for a spectrum of sophisticated functionalities, including remote target recognition and autonomous navigation and operations. In the forthcoming research, we aim to make a highly integrated environmental awareness system, enabling the robot to navigate diverse environments autonomously while executing complex tasks with finesse. Through such advancements, we envision a future where robotic systems

exhibit unparalleled versatility and adaptability, revolutionizing industries ranging from rescue to exploration.

**Supplementary Materials:** The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/drones8050208/s1>.

**Author Contributions:** Methodology, J.G.; Software, J.G.; Validation, Q.Z.; Investigation, M.C. and Y.L. (Yueyang Li); Writing—original draft, J.G.; Writing—review and editing, H.Z.; Project administration, Y.L. (Yibin Li); Funding acquisition, H.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by the National Key Research and Development Program of China, Grant No. 2022YFB4701503; in part by the National Natural Science Foundation of China under Grant No. 62073191; in part by the Shandong Provincial Key R&D Program (Major Scientific and Technological Innovation Project) under Grant No. 2019JZZY010441; in part by the Shandong Provincial Natural Science Foundation under Grant No. ZR2022MF296; and in part by the Project of the Shandong Province Higher Educational Youth and Innovation Talent Introduction and Education Program.

**Data Availability Statement:** Data are contained within the article and Supplementary Materials.

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

## References

- Chen, G.; Hong, L. Research on Environment Perception System of Quadruped Robots Based on LiDAR and Vision. *Drones* **2023**, *7*, 329. [CrossRef]
- Chen, T.; Li, Y.; Rong, X.; Zhang, G.; Chai, H.; Bi, J.; Wang, Q. Design and Control of a Novel Leg-Arm Multiplexing Mobile Operational Hexapod Robot. *IEEE Robot. Autom. Lett.* **2021**, *7*, 382–389. [CrossRef]
- Zhang, G.; Ma, S.; Shen, Y.; Li, Y. A Motion Planning Approach for Nonprehensile Manipulation and Locomotion Tasks of a Legged Robot. *IEEE Trans. Robot.* **2020**, *36*, 855–874. [CrossRef]
- Chen, T.; Sun, X.; Xu, Z.; Li, Y.; Rong, X.; Zhou, L. A trot and flying trot control method for quadruped robot based on optimal foot force distribution. *J. Bionic Eng.* **2019**, *16*, 621–632. [CrossRef]
- Chai, H.; Li, Y.; Song, R.; Zhang, G.; Zhang, Q.; Liu, S.; Hou, J.; Xin, Y.; Yuan, M.; Zhang, G.; et al. A survey of the development of quadruped robots: Joint configuration, dynamic locomotion control method and mobile manipulation approach. *Biomim. Intell. Robot.* **2022**, *2*, 100029. [CrossRef]
- Pang, L.; Cao, Z.; Yu, J.; Guan, P.; Rong, X.; Chai, H. A visual leader-following approach with a TDR framework for quadruped robots. *IEEE Trans. Syst. Man. Cybern. Syst.* **2019**, *51*, 2342–2354. [CrossRef]
- Wang, P.; Zhou, X.; Zhao, Q.; Wu, J.; Zhu, Q. Search-based Kinodynamic Motion Planning for Omnidirectional Quadruped Robots. In Proceedings of the 2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Virtually, 12–16 July 2021; pp. 823–829.
- Zhang, Y.; Tian, G.; Shao, X.; Liu, S.; Zhang, M.; Duan, P. Building metric-topological map to efficient object search for mobile robot. *IEEE Trans. Ind. Electron.* **2021**, *69*, 7076–7087. [CrossRef]
- Fu, X.; Wei, G.; Yuan, X.; Liang, Y.; Bo, Y. Efficient YOLOv7-Drone: An Enhanced Object Detection Approach for Drone Aerial Imagery. *Drones* **2023**, *7*, 616. [CrossRef]
- Miller, A.T.; Allen, P.K. Graspit! a versatile simulator for robotic grasping. *IEEE Robot. Autom. Mag.* **2004**, *11*, 110–122. [CrossRef]
- Pelossof, R.; Miller, A.; Allen, P.; Jebara, T. An SVM learning approach to robotic grasping. In Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA), New Orleans, LA, USA, 26 April–1 May 2004; Volume 4, pp. 3512–3518.
- Saxena, A.; Driemeyer, J.; Ng, A.Y. Robotic grasping of novel objects using vision. *Int. J. Robot. Res.* **2008**, *27*, 157–173. [CrossRef]
- Rusu, R.B.; Bradski, G.; Thibaux, R.; Hsu, J. Fast 3d recognition and pose using the viewpoint feature histogram. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 2155–2162.
- Lenz, I.; Lee, H.; Saxena, A. Deep learning for detecting robotic grasps. *Int. J. Robot. Res.* **2015**, *34*, 705–724. [CrossRef]
- Guo, D.; Sun, F.; Liu, H.; Kong, T.; Fang, B.; Xi, N. A hybrid deep architecture for robotic grasp detection. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1609–1614.
- Zhang, H.; Zhou, X.; Lan, X.; Li, J.; Tian, Z.; Zheng, N. A real-time robotic grasping approach with oriented anchor box. *IEEE Trans. Syst. Man. Cybern. Syst.* **2019**, *51*, 3014–3025. [CrossRef]
- 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 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 7223–7230.
- Chu, F.J.; Xu, R.; Vela, P.A. Real-world multiobject, multigrasp detection. *IEEE Robot. Autom. Lett.* **2018**, *3*, 3355–3362. [CrossRef]

19. 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.* **2022**, *41*, 690–705. [[CrossRef](#)]
20. Morrison, D.; Corke, P.; Leitner, J. Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach. *arXiv* **2018**, arXiv:1804.05172.
21. Ma, J.; Chen, P.; Xiong, X.; Zhang, L.; Yu, S.; Zhang, D. Research on Vision-Based Servoing and Trajectory Prediction Strategy for Capturing Illegal Drones. *Drones* **2024**, *8*, 127. [[CrossRef](#)]
22. Tranzatto, M.; Miki, T.; Dharmadhikari, M.; Bernreiter, L.; Kulkarni, M.; Mascari, F.; Andersson, O.; Khattak, S.; Hutter, M.; Siegwart, R.; et al. CERBERUS in the DARPA Subterranean Challenge. *Sci. Robot.* **2022**, *7*, eabp9742. [[CrossRef](#)]
23. Chen, H.; Chen, Y.; Wang, M. Trajectory tracking for underactuated surface vessels with time delays and unknown control directions. *IET Control Theory Appl.* **2022**, *16*, 587–599. [[CrossRef](#)]
24. Chen, H.; Shen, C.; Huang, J.; Cao, Y. Event-triggered model-free adaptive control for a class of surface vessels with time-delay and external disturbance via state observer. *J. Syst. Eng. Electron.* **2023**, *34*, 783–797. [[CrossRef](#)]
25. Chen, Y.; Chen, H. Prescribed performance control of underactuated surface vessels' trajectory using a neural network and integral time-delay sliding mode. *Kybernetika* **2023**, *59*, 273–293. [[CrossRef](#)]
26. Park, D.H.; Kwon, J.H.; Ha, I.J. Novel position-based visual servoing approach to robust global stability under field-of-view constraint. *IEEE Trans. Ind. Electron.* **2011**, *59*, 4735–4752. [[CrossRef](#)]
27. Shen, T.; Radmard, S.; Chan, A.; Croft, E.A.; Chesi, G. Optimized vision-based robot motion planning from multiple demonstrations. *Auton. Robot.* **2018**, *42*, 1117–1132. [[CrossRef](#)]
28. Shi, H.; Sun, G.; Wang, Y.; Hwang, K.S. Adaptive image-based visual servoing with temporary loss of the visual signal. *IEEE Trans. Ind. Inform.* **2018**, *15*, 1956–1965. [[CrossRef](#)]
29. Redmon, J.; Angelova, A. Real-time grasp detection using convolutional neural networks. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 1316–1322.
30. Wang, Z.; Li, Z.; Wang, B.; Liu, H. Robot grasp detection using multimodal deep convolutional neural networks. *Adv. Mech. Eng.* **2016**, *8*, 1687814016668077. [[CrossRef](#)]
31. Asif, U.; Tang, J.; Harrer, S. GraspNet: An Efficient Convolutional Neural Network for Real-time Grasp Detection for Low-powered Devices. In Proceedings of the 2018 International Joint Conference on Artificial Intelligence(IJCAI), Stockholm, Sweden, 13–19 July 2018; Volume 7, pp. 4875–4882.
32. 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.
33. Depierre, A.; Dellandréa, E.; Chen, L. Scoring Graspability based on Grasp Regression for Better Grasp Prediction. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 4370–4376. [[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.