

## Article

# Vision-Guided Hand–Eye Coordination for Robotic Grasping and Its Application in Tangram Puzzles

Hui Wei , Sicong Pan, Gang Ma and Xiao Duan

Laboratory of Algorithms for Cognitive Models, School of Computer Science and Technology, Fudan University, Shanghai 200433, China; 18210240033@fudan.edu.cn (S.P.); 18210240142@fudan.edu.cn (G.M.); 18210240074@fudan.edu.cn (X.D.)

\* Correspondence: weihui@fudan.edu.cn

**Abstract:** In this study we present an autonomous grasping system that uses a vision-guided hand–eye coordination policy with closed-loop vision-based control to ensure a sufficient task success rate while maintaining acceptable manipulation precision. When facing a diversity of tasks with complex environments, an autonomous robot should use the concept of task precision, including the accuracy of perception and precision of manipulation, as opposed to just the grasping success rate typically used in previous works. Task precision combines the advantages of grasping behaviors observed in humans and a grasping method applied in existing works. A visual servoing approach and a subtask decomposition strategy are proposed here to obtain the desired level of task precision. Our system performs satisfactorily on a tangram puzzle task. The experiments highlight the accuracy of perception, precision of manipulation, and robustness of the system. Moreover, the system is of great significance for improving the adaptability and flexibility of autonomous robots.

**Keywords:** robotic grasping; vision-guided; hand–eye coordination



**Citation:** Wei, H.; Pan, S.; Ma, G.; Duan, X. Vision-Guided Hand–Eye Coordination for Robotic Grasping and Its Application in Tangram Puzzles. *AI* **2021**, *2*, 209–228. <https://doi.org/10.3390/ai2020013>

Academic Editor: Emanuele Frontoni

Received: 9 March 2021

Accepted: 26 April 2021

Published: 4 May 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 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

In this study, we outline a general framework of robot control to solve a class of intelligent grasping tasks. For such tasks, the environment is variable, and so the framework of solving tasks cannot be defined in advance with a fixed sequence of motions. For example, in order for a home-service robot to accomplish a cup-carrying task, it must be able to adapt to changes in the color, lighting, position, shape, background, and other parameters of a cup and its surrounding environment. The robot needs to reliably recognize the target object, accurately locate the object, and have a closed-loop control process. These three requirements are difficult for an intelligent robot to satisfy because they present challenging computer vision task problems, such as invariant recognition under a complex background, optical measurement, and adaptive control with visual feedback.

In many existing works on robot grasping, the robot first perceives the scene and recognizes appropriate grasp locations, then plans a path to those locations, and finally follows the planned path to those locations [1,2]. These tasks are respectively called perception, planning, and action. However, the grasping behaviors observed in humans are dynamical processes that interleave sensing and manipulation at every module [3]. Robot hand–eye coordination is a typical feedback-based, closed-loop control process similar to grasping behaviors observed in humans. The core function of the process is to guide the motion of the robot based on the relative displacement between the target location and the end-effector of the robot [4,5]. This eliminates the work of calibration and transformation between multiple coordinate systems.

To achieve vision-guided hand–eye coordination, the core function must be redesigned to sense changes in both the environment and the target object and then provide a corresponding control strategy. We propose task precision as the core function, which can be calculated by the relative distance between the target object and the end-effector of the

robot. The system is expected to have the ability to (1) recognize both the end-effector of the robot and the target object (recognition ability), (2) accurately describe their contours to determine their relative displacement (locating ability), and (3) track the movement of the robot towards the end-effector and the target object (tracking ability).

Humans have the ability to precisely solve tangram puzzles (see Section 4.1 for more details on this task). However, since the game background and target layout are both changeable, it is difficult for a robot to achieve hand–eye coordination and solve tangram puzzles. The vision-guided hand–eye coordination system in this study can complete the autonomous decision-making process like a human can.

The paper is organized as follows. Related robotic grasping research is introduced in Section 2. The framework of the system is reported in Section 3. The application of the system to solving a tangram puzzle is described in Section 4. The experimental results are presented in Section 5, and the conclusion is given in Section 6. Our robot environment is described in Appendix A.

The main contributions of the work are as follows: (1) Vision-guided hand–eye coordination is analyzed, and a new concept of task precision in robot grasping tasks is proposed, taking inspiration from the original closed-loop hand–eye coordination method. (2) The perception, planning, and action stages in existing works are integrated into a module of our system so that we can choose different policies when facing different grasping tasks. (3) Our experimental evaluation demonstrates the effectiveness of this approach, which can run with only a laptop, in terms of both a sufficient task success rate and acceptable manipulation with precision.

## 2. Related Work

### 2.1. Hand–Eye Coordination

Hand–eye coordination can be divided into two main types, calibrated and uncalibrated. Hill and Park first achieved closed-loop control in a calibrated hand–eye coordination method [4]. Calibrated hand–eye coordination methods [4,6–8] obtain the corresponding relationship between the camera and the robot base by using the three-dimensional space transformation principle and the known camera parameters. This kind of method requires a lot of labor; the accuracy of the parameters depends on the calibration, and recalibration is necessary when camera parameters change. Therefore, this kind of method is not suitable for the manipulator environment with open operation.

Many studies have been conducted on the uncalibrated hand–eye coordination of robots. Herve [9] demonstrated the feasibility of mapping from image space to robot space. Meanwhile, Su et al. proposed an uncalibrated robot hand–eye coordination system that improved the robot’s environmental adaptability [5,10,11]. Levine et al. used deep learning in large-scale datasets to achieve hand–eye coordination [12,13]. The learning-based method uses multiple sensors and a large amount of pretraining data [14,15], while the other methods require a manual presentation of some of the first trial motions. Therefore, this type of approach is not suitable for open environment tasks. Although these methods simplify the challenging task of hand–eye coordination calibration and location, it becomes more challenging to guide the robot to make correct motion decisions on practical problems. For instance, if the goal of a robot with uncalibrated hand–eye coordination is to play chess, the robot needs to know where the pieces are placed before it can pick them up and place them in the correct position. The robot does not determine for itself where to place a piece.

### 2.2. Robotic Grasping

With the development of hardware technology, it has become possible to use mature three-dimensional laser ranging technology and point cloud processing to achieve three-dimensional reconstruction and to complete the task of robot grasping. Several works on this topic have used open-loop planners to determine the best location at which to grasp [1,2,16]. In contrast, our system uses vision-guided hand–eye coordination, which

enables closed-loop vision-based control. Therefore, our system can respond to dynamic disturbances and deal with complex environments.

In recent years, there have also been many studies on closed-loop grasping [12,17–28]. For example, the Google team proposed a vision-based deep reinforcement learning algorithm to realize robotic grasping, which enables closed-loop control [10]. The robot can grasp objects unknown to the model with a grasping success rate of 96%. These works all focus on the grasping success rate. However, most tasks faced by intelligent robots have hard constraints such as a variable environment, time sensitivity, and the need for precise manipulation, so only focusing on the grasping is not enough, as in the example of playing chess above. Moreover, solving a tangram puzzle requires not only grasp success (how to grasp) but also task success (how to rotate and put down). Our system, which uses task precision rather than the grasping success rate, has strong adaptability and intelligence for tasks and maintains acceptable manipulation precision.

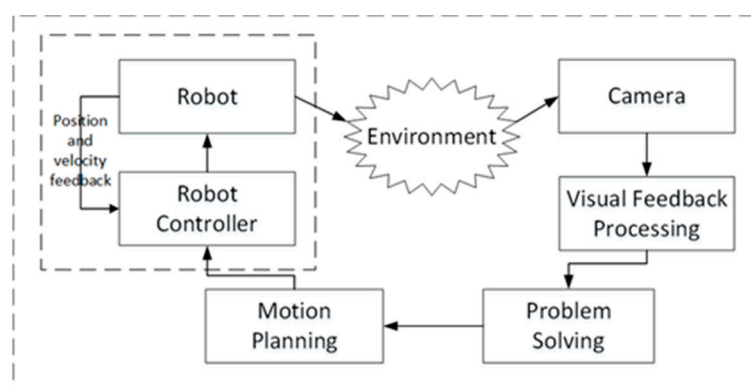
### 2.3. Visual Feedback

The classic choice for visual feedback is visual features. In recent years, deep learning technology has seen a great breakthrough in automatic feature extraction with the development of the convolutional neural network (CNN) to Faster R-CNN and Mask R-CNN [29–31]. These algorithms can effectively complete instance segmentation and can be used for “object detection” and “object key point detection”. For example, the algorithms can be used to identify the tangram and its contours in a 2D picture. These algorithms are usually evaluated by recall, precision, and pattern classification indicators. The precision attained so far, however, is far from sufficient to guide the robot through delicate movements. Moreover, the good performance of these algorithms is generally based on large datasets and sufficient machine learning training in advance, so the cost of data acquisition, manual annotation, high-performance hardware, and other aspects of the final product is very high. In contrast, our system uses simple contour-based methods that require no training, and it adapts to complex backgrounds with enough precision.

## 3. Vision-Guided Hand–Eye Coordination for Robotic Grasping

### 3.1. System Structure

In this study, we used a vision-guided hand–eye coordination system as shown in Figure 1. The camera first acquires images of the task environment. An image is extracted by the visual processing module. The problem-solving module confirms the current action according to the environmental information and determines the best task to carry out. The motion planning module guides the robot’s manipulation. The robot itself is composed of a controller and a body, forming a closed loop. Thus, the whole system forms a typical double closed-loop system.



**Figure 1.** The structure of the vision-guided hand–eye coordination system.

The robot and its controller belong to the hardware system. The hardware system includes robot hardware, kinematics, and communication systems. The hardware system is not the focus on this paper, but its structure is shown in Appendix A.

According to the principles of Sanderson and Weiss [32] and other related works, each module has multiple choices. The camera can be mounted on the arm of the robot (i.e., eye in hand) or somewhere in the environment (i.e., eye to hand). The visual processing module can have an RGB image or RGB-D image as the input and can select the feature and algorithm accordingly. The problem-solving module can use search, reasoning, or learning methods to achieve its function. The motion planning module can use a sampling-based approach (such as RRT) or a combinatorial approach.

### 3.2. Problem Solving

The robotic grasping task in our study has the following requirements: (1) good real-time performance, meaning the robot needs to respond to real-time changes in the environment; and (2) high manipulation precision, meaning that a large error leads to mission failure. Solving such a task requires a real-time, high-precision system, and we contend that the vision-guided robot hand–eye coordination system is the best choice.

---

#### Algorithm 1 Servoing

---

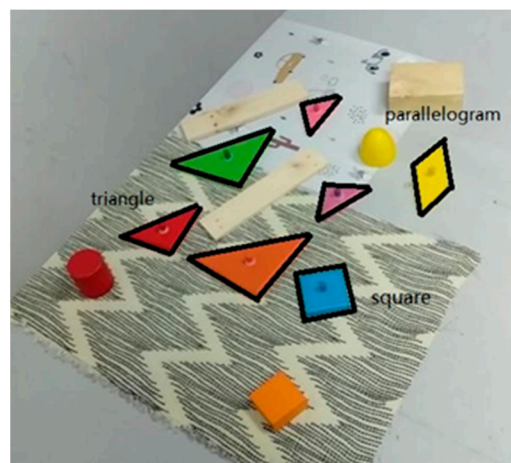
- 1: Given current image  $X$  and Task Precision  $f(t_i, p)$
  - 2: Get state  $s$  from visual processing module
  - 3: Calculate  $p$  with image  $X$  and robot state
  - 4: Get subtask  $t_i$  from problem-solving module and  $s$
  - 5:  $n = f(t_i, p)$
  - 6: for  $1 \dots n$  do
  - 7: Execute  $t_i$  with motion planning module
- 

Similar to hierarchical reinforcement learning [33], we propose having the whole task to be composed of several subtasks  $t \in T$  and each subtask to be composed of several actions  $a \in A$ . The key to problem solving is the Task Precision function  $f(t_i, p)$ , which uses input subtasks  $t_i$  and precision  $p$  to obtain the number of dynamic divisions  $n$ . The precision  $p$  can be defined as the relative distance between the target object and the end-effector of the robot. The Task Precision function can be predefined or learned. Finally, we propose Algorithm 1 Servoing to guide the robot with continuous control.

## 4. Application to Completing Tangram Puzzle

### 4.1. Task Description

The tangram puzzle is a toy composed of seven blocks: five isosceles right triangles, a square, and a parallelogram [34]. The puzzle is shown in Figure 2.



**Figure 2.** The tangram blocks that need to be operated on in the task.



In this study, our system framework is applied to the tangram puzzle task. The main process of the task is as follows. (1) Tangram blocks are laid down on the table randomly. (2) The target state image with a given pattern (see in Section 4.4) is known. (3) The robot selects the tangram blocks in the needed order and lays them down in accurate positions (see Section 4.5.4 for more detail).

As we are primarily concerned with our system having acceptable precision of manipulation, for this task, we can make the following assumptions: (1) the objects have one solid color and are not textured, and (2) the offset from the camera center to gripper center is known. We use preprogrammed motion to compensate for this offset.

#### 4.2. Vision-Guided Hand–Eye Coordination for Tangram Task

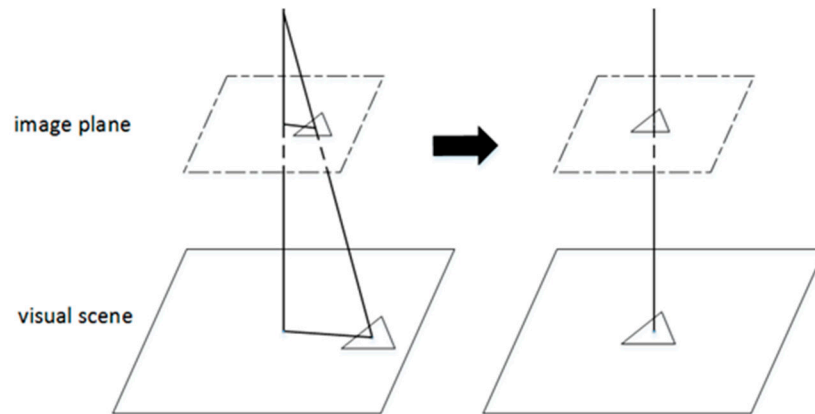
In our system, the monocular camera for RGB imaging is mounted on the arm of the robot. We use the search policy in the problem-solving module and the combinatorial approach in the motion planning module.

The tangram task can be regarded as a 2D planar grasp, which means that the target object lies on a planar workspace and the grasp is constrained from one direction. In this case, the height of the gripper is fixed, and the gripper direction is perpendicular to one plane. Therefore, the essential information is simplified from 6D to 3D, including the 2D in-plane positions and the 1D rotation angle.

The key to solving the task lies in understanding the environment. We selected the centers of the shapes as image features. As the offset from the camera center to the gripper center is known, the error between the center of the shape and the center of the two-dimensional image (see in Section 4.5.1) can be used to calculate  $p$  instead of the relative distance between the target object and the end-effector of the robot, as shown in Figure 3.

Additionally, we predefined Task Precision  $f(t_i, p)$  as follows:

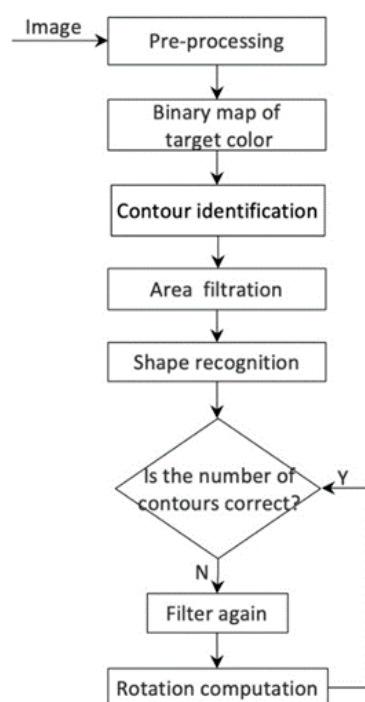
$$f(t_i, p) = \begin{cases} 3 & t_i = \text{Moveto}(X) \\ 1 & \text{other} \end{cases} \quad (1)$$



**Figure 3.** Process illustration of visual feedback.

#### 4.3. Visual Processing of the Tangram Puzzle

The complex background forces the system to perform effective preprocessing on the images. To ensure that the system can recognize and extract features of images such as shape, position, and pose, the shape recognition and rotation calculations are added to the visual processing module. Figure 4 illustrates the stages of precise object recognition.

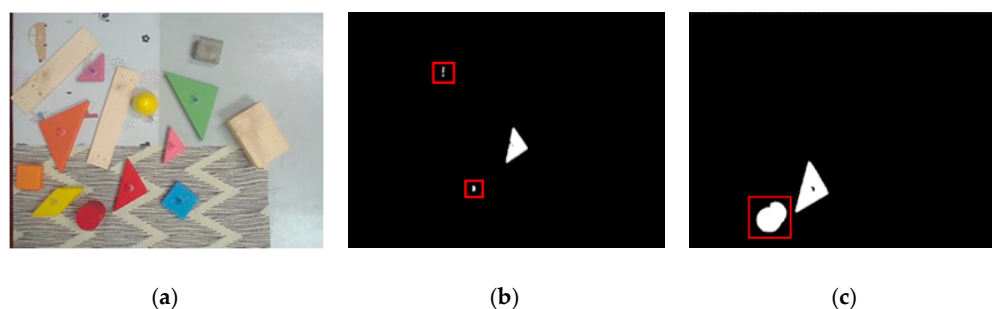


**Figure 4.** Flowchart of image processing.

#### 4.3.1. Shape Recognition

In considering (1) the lack of a large number of training samples, (2) the requirement of minimal computing resources and fast computing speed, and (3) the needs of fine-grained image analysis, the system adopted the basic image processing method and mathematical judgment instead of using the neural network. First, simple image enhancement operations are applied to the acquired image, such as Gaussian de-noising, brightness, and contrast adjustment [35]. To filter out different blocks according to color, the system converts the image mode from RGB to HSV for a more favorable segmentation interval. Simultaneously, morphological expansion and erosion operations are used to optimize regional boundaries [35].

The obtained results were not satisfying because of the complexity of the background and the limitations of the color filtering algorithm. The system needed to remove the noise that disturbs the task, and this process runs through the entire shape recognition process. First, small areas (e.g., the areas in the red boxes of Figure 5b) can be filtered out by area computation, and the remaining parts are distinguished by shape, as shown in Figure 5c.



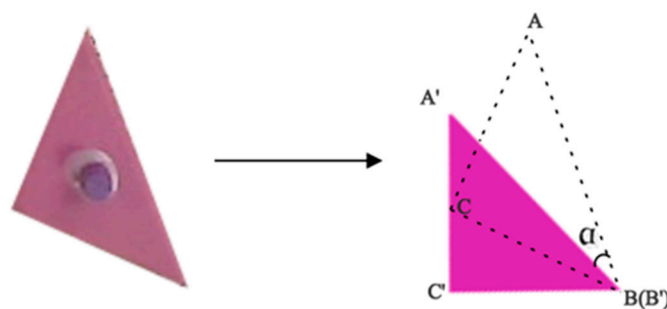
**Figure 5.** Preliminary processing: (a) the image captured by the camera; (b) the binary map of the pink block; (c) the binary map of red blocks, including a cylinder and a triangle.

The system determines the vertices of each area (e.g., the areas in Figure 5c) by approximating its contour. According to the number of vertices, the areas are divided into

two types. Then, the triangle is identified according to the angles and the relationships between the three sides. The square and the parallelogram can be distinguished by the aspect ratio, but the parallelogram needs to be further determined by finding out whether the opposite sides are parallel.

#### 4.3.2. Rotation Computation

Seven tangram blocks are placed on the table with random positions and poses. To put the messy blocks into a preconceived pattern (i.e., a target state image), the system needs to recognize the current posture of each block and rotate the blocks to fit the target state image. Based on the recognition process in Section 4.3.1, the rotation angle  $\alpha$  can be calculated according to the detected image and the target state image, as shown in Figure 6.



**Figure 6.** Calculation of the rotation angle.

The difficulty of rotation computation lies in the accuracy of mathematical calculations, which depends on the accuracy of recognition of contours and corners. Therefore, the entire process requires detailed and accurate information to be extracted from the image. Accuracy here reduces the error of positioning in the later stage of the task.

#### 4.4. Tangram Problem Solving

Our system uses the search algorithm to solve the problem and find the target state in the state space [36]. Newell and Simon put forward the means–ends analysis method in 1961 [37]. By analyzing the actions required by the environmental state to be achieved and the influence of actions on the environmental state, search-based problem solving can be realized.

To solve the problem of this study, we can use the means–ends analysis method, which is composed of the subtasks of the robot [38]. We do not need to care about microlevel details or details about the hardware. Instead, we can define subtasks at a higher level, such as the impact of actions on the task environment. At the same time, we need to describe whether a particular action changes the task environment and describe the state of the environment with a series of predicates.

Based on this concept, we can use an approach called STRIPS [39], which uses three action elements (i.e., premise, add list, delete list) to represent the mission planning process. The first element is the set of premises (P), which must be satisfied when an operation is applied. The second element is the add list (A), which applies an operation to add the state of the environment. The third element is the delete list (D), which applies an action to reduce the state of the environment. Such a series of actions can be organized within a triangular table.

The environment state is described with a series of predicates:

*gripping*(X)—The robot holds a tangram X in its hand

*ontable*(X)—Tangram block X is on the table

*located*(X)—Tangram block X's position is known

*rotated*(X)—Tangram block X has been rotated around the specified angle

*inplace*(X)—Tangram block X is in the specified position

*havepattern*()—Information about the pattern is obtained

*camera()*—Camera ready

The subtasks (bold) performed by the robot include the following:

**Moveto(X)**—Locate and move to the tangram block

**Pickup(X)**—Grab the tangram block

**Rotate(X)**—Rotate the tangram block

**Putdown(X)**—Put down the tangram block

**Evaluate(X)**—Check whether all tangram blocks, including X, are in the specified position

In practice, the subtask may consist of a series of actions, such as moving to a specific position and rotating to a specific angle (see Section 4.5). Such a design can correspond to the interface functions provided by the Step 700E robot (see Appendix A.1) and allow the robot to realize the change in environmental state.

Define the initial state of the tangram task as  $ontable(X_i) \wedge gripping() \wedge camera()$ , where  $X_i$  corresponds to the seven pieces of the tangram puzzle  $i = 1 - 7$ .

The completed status is  $inplace(X_i)$ .

In Figure 7, there is no set of premises for the total task **Tangram()**, which indicates the start of the task. Arrow means branch structures (i.e., move to a different action). In **Nextpuzzle()**, True indicates the presence of a disassembled tangram block and **Nextpuzzle()** moves to **Place(X)**, whereas False ends the task. In **Evaluate(X)**, True means a correct placement, whereas False means failure in the placement, and **Evaluate(X)** changes the environment state to  $ontable(X)$ .

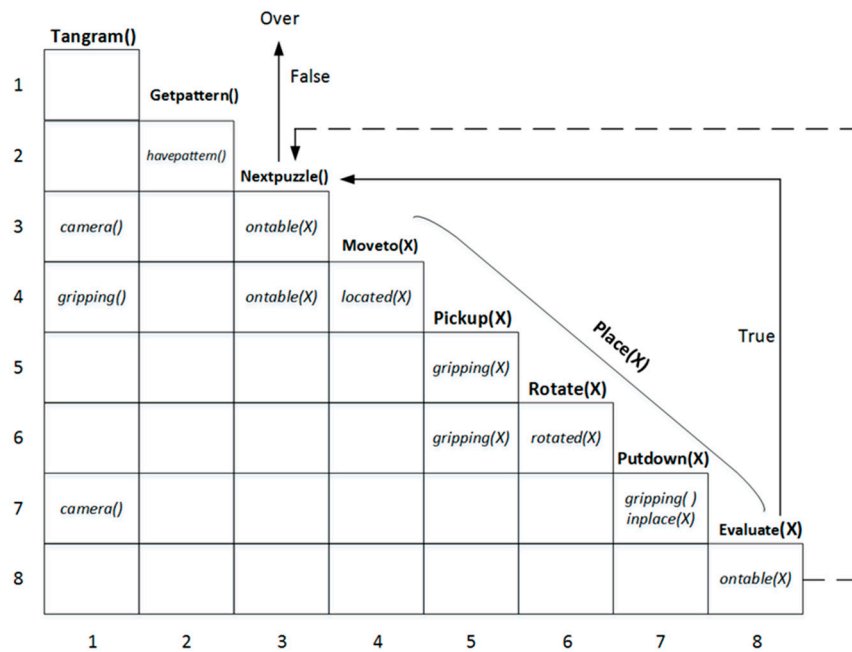
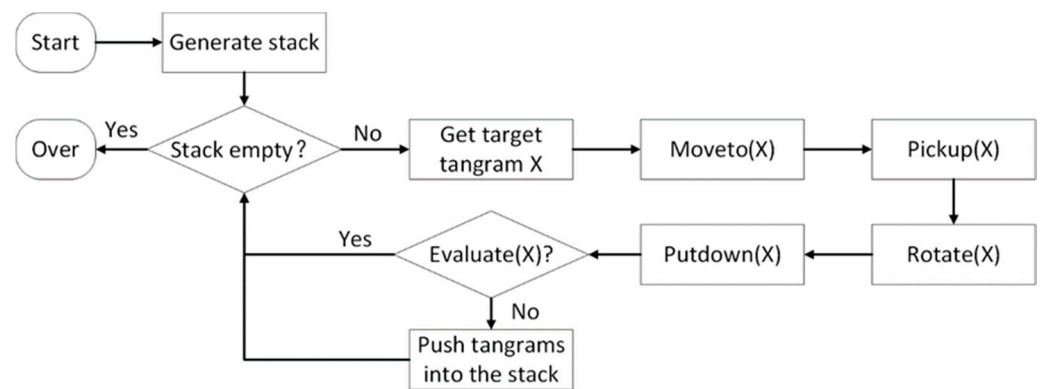


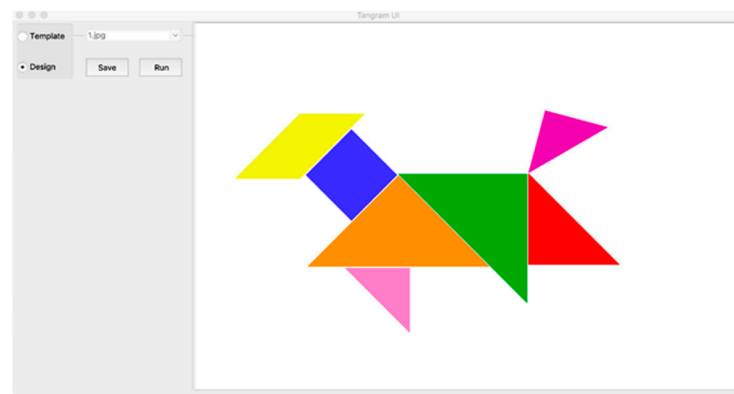
Figure 7. Tangram triangle table.

With the aid of the triangle table, a flowchart for solving the tangram puzzle can be drawn (see Figure 8). The implementation of  $ontable(X)$  uses a stack to determine whether the tangram block is on the desktop.



**Figure 8.** Tangram problem-solving flowchart.

Before motion planning, the system analyzes the target state image, which is designed by us in the client. Figure 9 illustrates an example of this (i.e., a dog pattern).



**Figure 9.** An image with a dog pattern.

The system uses the shape recognition algorithm introduced in Section 4.3.1 to process the image of the target state to obtain the color, shape, and coordinates of each tangram block's center. After sorting the coordinates of the tangram block's center as needed, the robot places each tangram block.

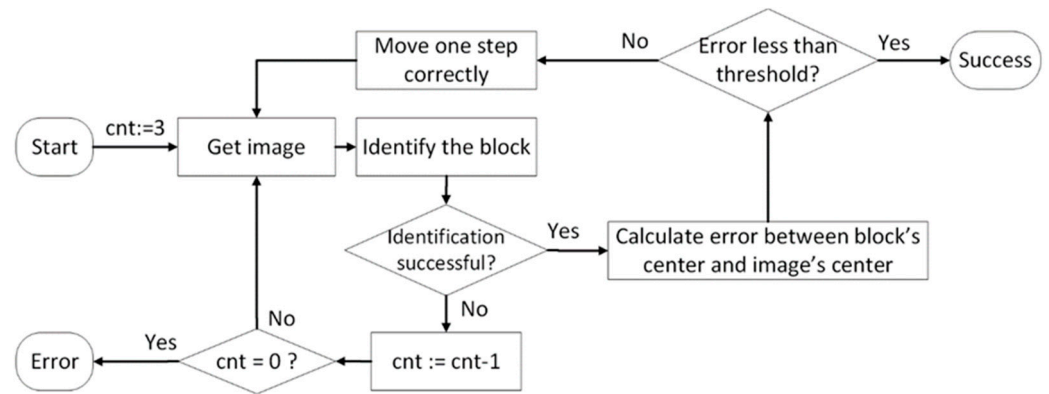
#### 4.5. Motion Planning

The motion planning module determines how to execute the atomic actions of the robot. For instance, locating the tangram blocks requires the robot to perform multiple moves, as discussed in the following subsection.

##### 4.5.1. Locating the Tangram Blocks

The main task of the locating operation is to obtain the error between the image's center and the tangram block's center by using the visual feedback module so that the robot can accurately differentiate a specified tangram block from the randomly placed blocks. Figure 10 shows the algorithm flow of this task.

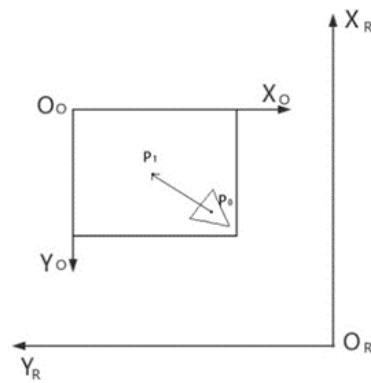




**Figure 10.** Flowchart of the locating operation.

In Figure 10, *cnt* is the user-defined count number of fault tolerances, and its initial value is 3. If the camera captures an image three times and the system still cannot correctly identify the tangram block with the specified color and shape, an error is reported.

After correctly identifying the tangram block, the system determines whether the tangram block is in the center of the camera's field of view by  $\vec{P_0P_1}$  ( $x_1 - x_0, y_1 - y_0$ ), where  $P_0(x_0, y_0)$  is the coordinate of the tangram block's center in the image, and  $P_1(x_1, y_1)$  is the coordinate of the image's center. The coordinates are shown in Figure 11. The system defines a threshold as  $thresh(pix)$ . When  $|x_1 - x_0| < thresh$  and  $|y_1 - y_0| < thresh$ , the system considers the block to be in the center of the camera's field of view. Otherwise, the system calculates the correct direction of movement from  $\vec{P_0P_1}$  according to the relationship between the world coordinates of the robot and the image coordinates. Then, the end-effector moves one step in the correct direction to gradually approach the target block. The value of *thresh* and the size of the step are reduced adaptively with the height of the robot's end-effector.



**Figure 11.** The relationship between the world coordinate ( $X_R O_R Y_R$ ) of the robot and the image coordinate ( $X_O O_O Y_O$ ) on a two-dimensional plane.

#### 4.5.2. Picking up Tangram Blocks

After locating a specified block, the block is in the center of the camera's field of view. In engaging hand-eye coordination, the end-effector of the robot is moved to the grasping position. Then, the block is picked up by the electric gripper.

#### 4.5.3. Rotating Tangram Blocks

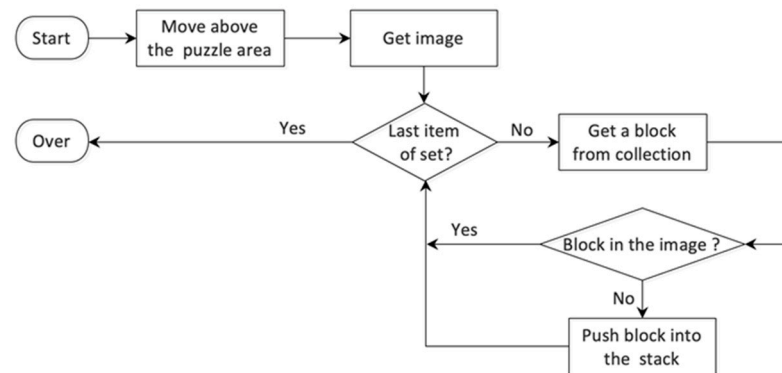
After correctly locating and grabbing the tangram block, we focus on how to accurately place it. First, the system calculates the rotation angle of the block by using the rotation algorithm introduced in Section 4.3.2. Then, the robot uses its end-effector to rotate the block to adjust the block's pose.

#### 4.5.4. Putting down Tangram Blocks

This section describes how to calculate the position of tangram blocks in the planar world coordinate of the robot. We refer to the images with special patterns (see Figure 9) as goal state images. By proportionally magnifying the length and width of the goal state image, the system sets a puzzle area to put down tangram blocks in the robot's planar world coordinate. The coordinate value of the tangram block's center in the world coordinate is calculated according to the pixel coordinate values of the tangram block's center in the goal state image. Since the initial height of the robot's end-effector is known, after obtaining the coordinate value of the block's center, the robot can accurately put down blocks in the puzzle area. Finally, all blocks are placed in the pattern indicated in the goal state image.

#### 4.5.5. Evaluating Tangram Blocks

The evaluation section provides a function with which to check the tangram blocks placed by the robot (shown in Figure 12). The system defines a set to record the placed blocks. It checks whether each tangram block recorded in the collection is at the specified puzzle area. If not, the tangram block is pushed into the stack.



**Figure 12.** Flowchart of evaluation.

## 5. Experiments

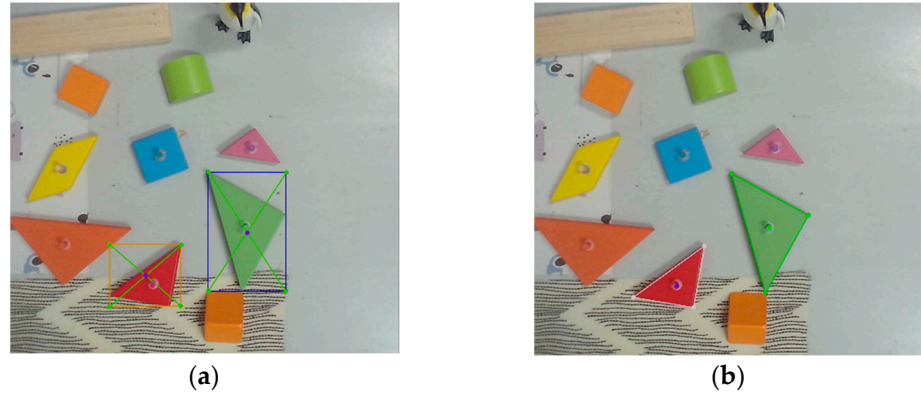
### 5.1. Visual Feedback Statistical Experiment

To demonstrate the effectiveness of our visual feedback processing algorithm, we compare its performance with the widely used visual feedback algorithms (Faster R-CNN [21] and Mask R-CNN [22]). In our visual feedback experiment, the background is complex, while the position and pose of the object are changeable.

#### 5.1.1. Visual Feedback Indicators

The square error in a picture is defined as  $E_{square} = (\sum_{i=1}^n (m_i - gt_i)^2) / n$ , where  $m_i$  is the position of the center of mass of the  $i$ -th tangram on the picture and  $gt_i$  is the ground truth position of the center of mass of the  $i$ -th tangram. In other words, we evaluate the accuracy of the visual feedback by calculating the center of mass point deviation from each tangram.

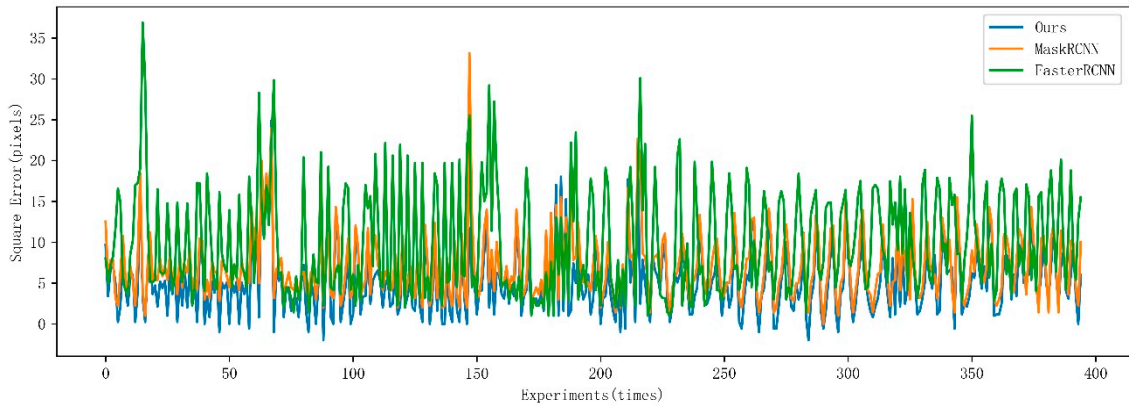
In Figure 13, we provide a visual look at the square error. The Mask R-CNN algorithm predicts the position of each tangram and obtains the mask of the object, based on which mass center of each tangram can be obtained. Our algorithm predicts the center of mass from the contour of each tangram.



**Figure 13.** Visual feedback algorithm evaluation. (a) The center of mass obtained by the masking method. (b) The center of mass obtained by the contour method.

### 5.1.2. Visual Feedback Statistical Results

We count the square errors of 400 pictures and take the average error of all the blocks of each picture as an experimental result. As shown in Figure 14, our result is generally equal to or better than that of the comparison algorithm.



**Figure 14.** Visual feedback experiments.

## 5.2. Tangram Experiment

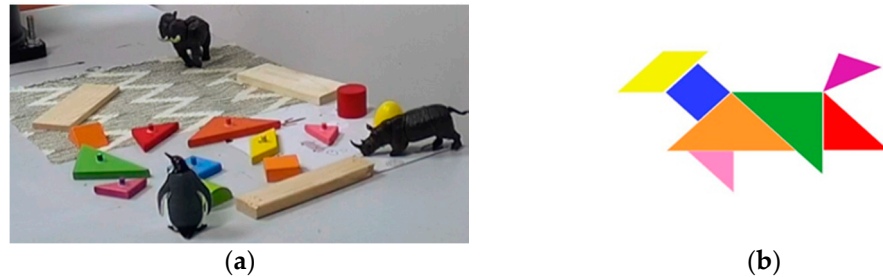
The tangram blocks are placed in any position and pose in a complex background. Then, the robot makes reasonable decisions to complete the tangram puzzle based on the target state randomly assigned by us. During the execution of the task, we can change the task environment at will. For example, we can verify whether the robot can still make correct decisions when we remove a block that had been placed previously or is currently being identified.

### 5.2.1. Tangram Indicators

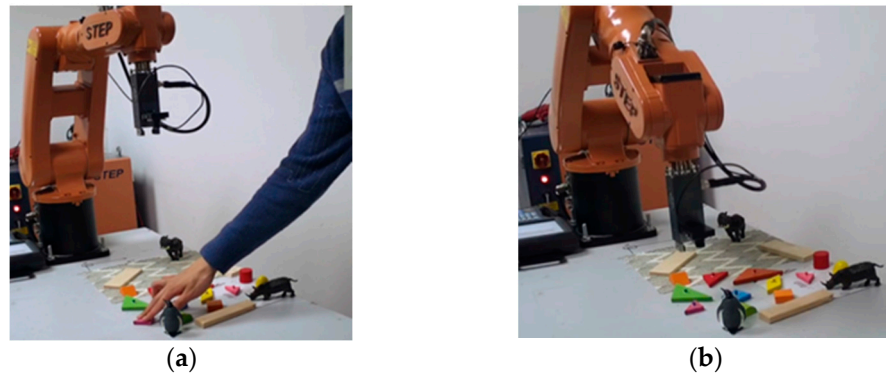
The gap error between tangram blocks is defined as  $E_{gap} = (\sum |d_{ij} - r_{ij}|) / 21$ , where  $0 < i < j < 8$ ,  $d_{ij}$  is the value of the distance between the mass centers of any two tangram blocks placed by the robot after finishing the tangram puzzle, and  $r_{ij}$  is the value of the distance between any two center points of the tangram blocks in the target state image. The difference between the two values is used to reflect the gap between the two tangram blocks. In this experiment, one pixel corresponds to  $70/256 \approx 0.273$  mm in the real environment.

### 5.2.2. Dog Pattern Experiment

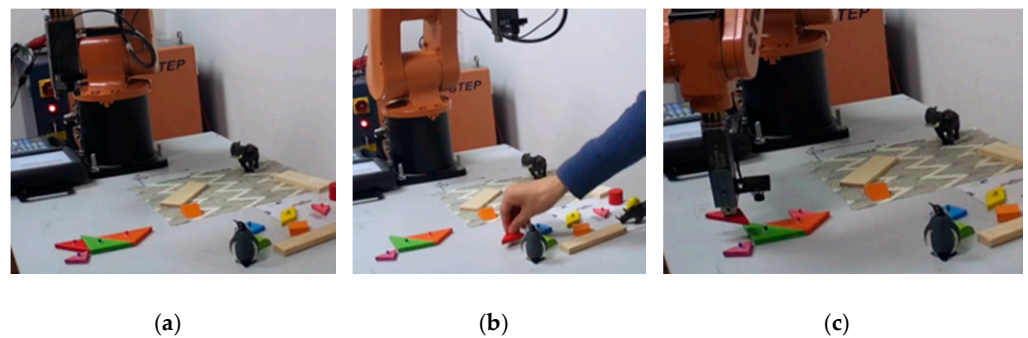
One of our examples, as shown in Figures 15–17, consists of an input state, target state, and our subtasks. The experimental indicators of this example are shown in Table 1. We list all the gap errors between any two tangram blocks.



**Figure 15.** The experiment input: (a) the tangram pieces randomly placed in the complex background; (b) an image with a dog pattern.






**Figure 16.** Recognition stage: (a) the tangram recognition process with the experimenter changing the task environment; (b) the process of grabbing after recognition.



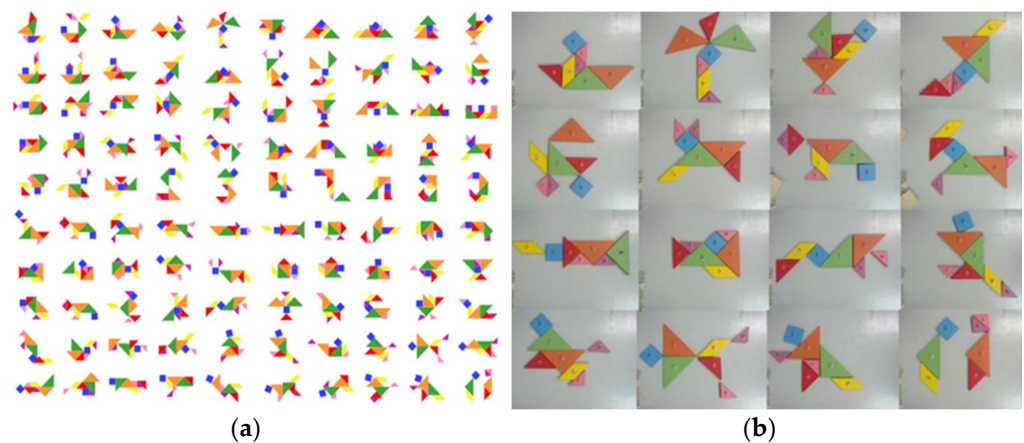
**Figure 17.** Placing tangram blocks. (a) A tangram block in task execution. (b) The red tangram block is removed. (c) After executing the current task, the red block task is detected to have failed and the program backtracks.

**Table 1.** Experimental data of three trials.

Target State Gap Error(mm)			
pink-red	1.03	0.94	0.92
pink-orange	0.42	0.43	1.04
pink-yellow	0.73	0.09	0.26
pink-green	0.81	0.06	0.06
pink-blue	0.43	1.32	0.68
pink-purple	1.45	1.04	0.08
red-orange	0.38	0.36	0.58
red-yellow	1.06	0.88	1.14
red-green	1.28	0.13	1.23
red-blue	0.60	0.35	0.27
red-purple	2.74	0.11	0.3
orange-yellow	1.04	0.13	1.04
orange-green	0.94	0.6	1.57
orange-blue	0.78	0.7	1.17
orange-purple	1.78	0.39	0.02
yellow-green	0.47	0.23	0.05
yellow-blue	0.31	1.31	0.39
yellow-purple	2.03	0.59	0.26
green-blue	0.8	0.53	0.45
green-purple	2.02	0.04	0.81
blue-purple	1.77	0.48	0.46
Average	1.09	0.51	0.61
Standard deviation	0.64	0.39	0.45

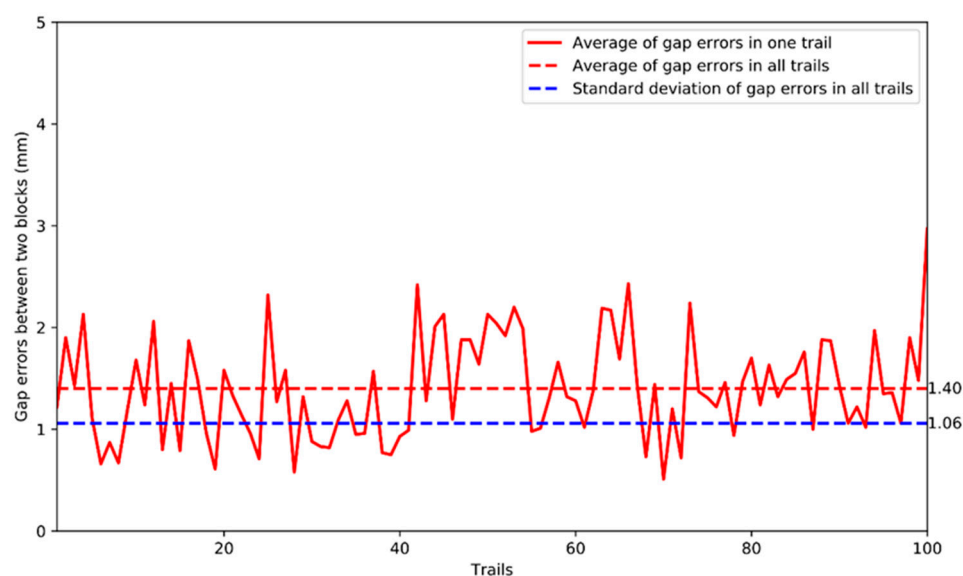
### 5.2.3. Statistical Experiment

A large number of experiments are used to test the performance of the proposed hand-eye coordination system for the tangram task. The target state images used are given in Figure 18, and the errors are given in Figure 19.



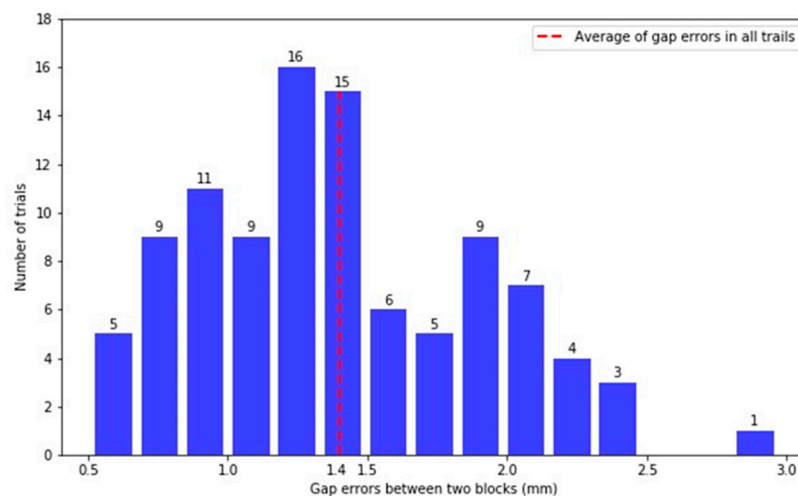
**Figure 18.** Experimental data: (a) 100 target state images with different patterns; (b) 16 sets of experimental results.





**Figure 19.** Statistics of error results of 100 trials.

In Figure 20, the solid red line represents the average gap error in one trial. The dotted red line shows the total average error; it is generally small, and the task accuracy is high. The dotted blue line is the total standard deviation, which shows that the error does not fluctuate very much.



**Figure 20.** Histogram of error distribution of 100 trials.

For the 100 trials, the number of trials is counted according to a certain range, indicating the error distribution. Figure 18 shows that most trials are concentrated near the average value, and 99% of the trials are concentrated within the range of 0.5–2.5 mm. The data are normally distributed. Therefore, the entire experiment is dominated by accidental errors [40].

## 6. Conclusions

This study presents a vision-guided hand–eye coordination system for robotic grasping. Our system is a lightweight, intelligent decision-making system utilizing complete closed-loop control. Owing to the implementation of the concept of task precision, the robot was able to complete some complex tasks through hand–eye coordination in a dynamic and changing environment and achieve high precision similarly to how humans can.

In this study, the robust system showed high adaptability and fault tolerance to the tangram task. This is helpful for promoting research on intelligent tasks such as cup-carrying by a household-service robot.

The hand–eye coordination system in this study needs to know the hand–eye relationship and the distance between the end-effector and the operating surface in advance. In order to improve intelligence, the system’s dependence on prior knowledge should be reduced to adapt to various hand–eye relationships. This is a possible direction for follow-up research.

**Author Contributions:** Conceptualization, H.W.; methodology, H.W., S.P., G.M., and X.D.; software, H.W., S.P., G.M., and X.D.; writing—original draft preparation, H.W., S.P., G.M., and X.D.; writing—review and editing, H.W., S.P., G.M., and X.D.; visualization, H.W., S.P., G.M., and X.D.; supervision, H.W.; project administration, H.W.; funding acquisition, H.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the NSFC Project [Project Nos. 61771146 and 61375122].

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

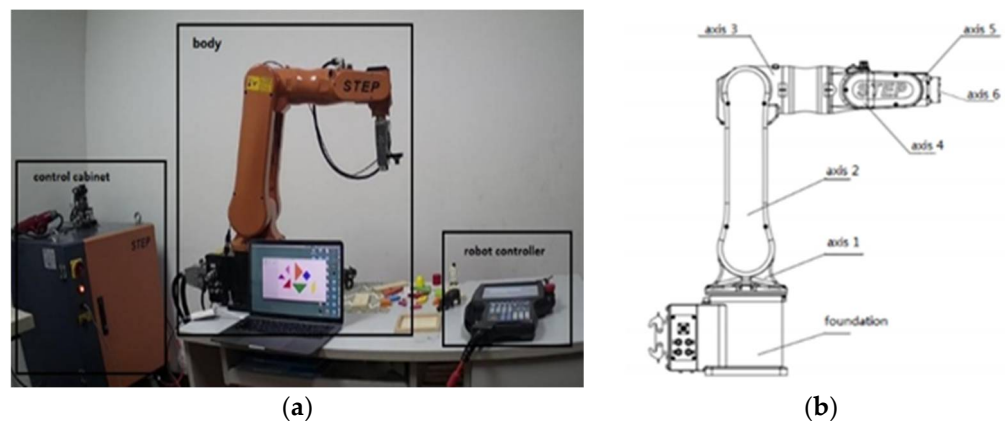
**Data Availability Statement:** All data included in this study are available upon request by contact with the corresponding author.

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

## Appendix A

### Appendix A.1. Robot Hardware

The STEP robot hardware system is composed of a body, robot controller, and control cabinet, as shown in Figure A1a. The body consists of six axes through which the robot moves, as shown in Figure A1b. The robot controller has basic software to control the robot operation with a series of control functions. The control cabinet contains the control hardware of the six axes, and it loads the control program written by CodeSys. The control program is the intermediary between the user and the robot controller.



**Figure A1.** STEP robot hardware system: (a) STEP robot environment; (b) STEP robot hardware.

### Appendix A.2. Kinematic Modeling

In this system, because an electric gripper was added, the connecting rod parameters of the robot needed to be added to the kinematics equation. Thus, the electric gripper can be regarded as an ideal Cartesian element.

According to the improved D-H representation, the transformation matrix from frame  $\{i-1\}$  to frame  $\{i\}$  can be written as follows:

$${}^{i-1}_i T Rot(x, \alpha_{i-1}) Trans(a_{i-1}, 0, 0) Rot(z, \theta_i) Trans(0, 0, d_i), \quad (A1)$$

where the parameters are shown in Table A1.

**Table A1.** Link parameters of SD700E robot.

$i$	$a_{i-1}$	$\alpha_{i-1}$	$d_i$	$\theta_i$
1	0	0	base	0
2	0	$-\pi/2$	0	$-\pi/2$
3	$l_2$	0	0	0
4	0	$-\pi/2$	$l_3$	0
5	0	$\pi/2$	0	0
6	0	$-\pi/2$	$l_5$	0

The general formula of the transformation matrix is as follows:

$${}^{i-1}_iT = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (\text{A2})$$

where  $c\theta_i$  is the cosine of  $\theta_i$ , which is abbreviated as  $c_i$  below, and  $s\theta_i$  is the sine of  $\theta_i$ , which is abbreviated as  $s_i$  below.

The relationship between the end of the mechanical arm and the base  ${}^0_6T$  is

$${}^0_6T = {}^0_1T(\theta_1) {}^1_2T(\theta_2) {}^2_3T(\theta_3) {}^3_4T(\theta_4) {}^4_5T(\theta_5) {}^5_6T(\theta_6), \quad (\text{A3})$$

According to the general formula, the transformation matrix of each connecting rod can be obtained as follows:

$${}^0_1T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & base \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (\text{A4})$$

$${}^1_2T = \begin{bmatrix} s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (\text{A5})$$

$${}^2_3T = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & l_2 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (\text{A6})$$

$${}^3_4T = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & 0 \\ 0 & 0 & 1 & l_3 \\ -s\theta_4 & -c\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (\text{A7})$$

$${}^4_5T = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_5 & c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (\text{A8})$$

$${}^5_6T = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ -s\theta_6 & -c\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (\text{A9})$$

$${}^0_6T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (A10)$$

where

$$\begin{cases} n_x = c_1[s_{23}(c_4c_5c_6 - s_4s_6) + c_{23}s_5c_6] + s_1(s_4c_5c_6 + c_4s_6) \\ n_y = s_1[s_{23}(c_4c_5c_6 - s_4s_6) + c_{23}s_5c_6] - c_1(s_4c_5c_6 + c_4s_6) \\ n_z = c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6 \\ o_x = -c_1[s_{23}(c_4c_5s_6 + s_4c_6) + c_{23}s_5s_6] - s_1(s_4c_5s_6 - c_4c_6) \\ o_y = -s_1[s_{23}(c_4c_5s_6 + s_4c_6) + c_{23}s_5s_6] + c_1(s_4c_5s_6 - c_4c_6) \\ o_z = -c_{23}(c_4c_5s_6 + s_4c_6) + s_{23}s_5s_6 \\ a_x = c_1(-s_{23}c_4s_5 + c_{23}c_5) - s_1s_4s_5 \\ a_y = s_1(-s_{23}c_4s_5 + c_{23}c_5) + c_1s_4s_5 \\ a_z = -c_{23}c_4s_5 - s_{23}c_5 \\ p_x = c_1[(-s_{23}c_4s_5 + c_{23}c_5)l_5 + c_{23}l_3 + s_2l_2] - s_1s_4s_5l_5 \\ p_y = s_1[(-s_{23}c_4s_5 + c_{23}c_5)l_5 + c_{23}l_3 + s_2l_2] + c_1s_4s_5l_5 \\ p_z = -(c_{23}c_4s_5 + s_{23}c_5)l_5 - s_{23}l_3 + c_2l_2 + base \end{cases}, \quad (A11)$$

Here,  $s_{23}$  is  $\sin(\theta_2 + \theta_3)$ , and  $c_{23}$  is  $\cos(\theta_2 + \theta_3)$ .

In the reverse solving process, using the inverse transformation  ${}^0_6T$ , we can obtain:

$$\theta_1 = \text{atan2}(p_y - l_5a_y, p_x - l_5a_x) \quad (A12)$$

or

$$\theta_1 = \text{atan2}(l_5a_y - p_y, l_5a_x - p_x), \quad (A13)$$

$$\theta_2 = \text{atan2}(p_z - b - l_5a_z, (c_1a_x + s_1a_y)l_5 - c_1p_x - s_1p_y) - \text{atan2}\left(\frac{l_2 - s_3l_3}{\rho}, \pm \sqrt{1 - \left(\frac{l_2 - s_3l_3}{\rho}\right)^2}\right) \quad (A14)$$

where

$$s_3 = \frac{l_3^2 + l_2^2 - [-(c_1a_x + s_1a_y)l_5 + c_1p_x + s_1p_y]^2 - (-a_zl_5 + p_z - b)^2}{2l_2l_3}, \quad (A15)$$

$$\theta_3 = \text{atan2}\left(\frac{c_2l_2 + a_zl_5 - p_z + b}{l_3}, \frac{-(c_1a_x + s_1a_y)l_5 + c_1p_x + s_1p_y - s_2l_2}{l_3}\right) - \theta_2 \quad (A16)$$

$$\theta_4 = \text{atan2}(-s_1a_x + c_1a_y, -[s_{23}(c_1a_x + s_1a_y) + c_{23}a_z]) \quad (A17)$$

$$\theta_4 = \text{atan2}(s_1a_x - c_1a_y, s_{23}(c_1a_x + s_1a_y) + c_{23}a_z) \quad (A18)$$

$$\theta_5 = \text{atan2}(s_4(-s_1a_x + c_1a_y) - c_4[s_{23}(c_1a_x + s_1a_y) + c_{23}a_z], c_{23}(c_1a_x + s_1a_y) - s_{23}a_z) \quad (A19)$$

$$\theta_6 = \text{atan2}(-s_4[s_{23}(c_1n_x + s_1n_y) + c_{23}n_z] - c_4(-s_1n_x + c_1n_y), -s_4[s_{23}(c_1o_x + s_1o_y) + c_{23}o_z] - c_4(-s_1o_x + c_1o_y)) \quad (A20)$$

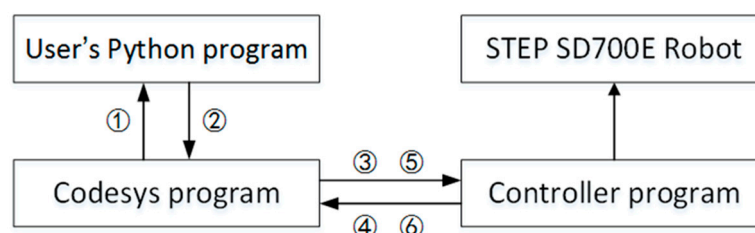
### A.3. Sequential Instruction Communication Protocol

To solve the transmission problem of the sequential motion instruction of the robot on the half-duplex port, we adopted a sequential instruction communication protocol based on a semaphore. Six steps complete the motion command communication and execution. Send and Finish are used to control the two semaphores between CodeSys and the robot controller program, where Send indicates whether new instructions were sent and Finish indicates whether the robot completed an action. The CodeSys program communicates with the user's Python program through a socket and with the robot controller program through a port.

Steps (shown in Figure A2.)

Send and Finish have initial values of F. Send is sent on port 4 and Finish on port 5.

1. When the CodeSys program detects that Send = F and Finish = F, it sends “ready” to the Python program.
  2. When Python receives the ready signal, it sends data to CodeSys.
  3. CodeSys receives the data sent by the Python program and then parses the data and passes it to the robot controller program. Set Send to T and write to port 4.
  4. When the robot controller detects Send = T and Finish = F, after executing the movement command, set Finish to T and write to port 5.
  5. When CodeSys detects Send = T and Finish = T, set Send to F and write to port 4.
  6. When the robot controller detects Send = F and Finish = T, set Finish to F and write to port 5.
- Return to step 1.



**Figure A2.** Illustration of the sequential instruction communication.

## References

1. Morrison, D.; Tow, A.W.; McTaggart, M.; Smith, R.; Kelly-Boxall, N.; Wade-McCue, S.; Erskine, J.; Grinover, R.; Gurman, A.; Hunn, T.; et al. Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–26 May 2018; pp. 7757–7764.
2. Ten Pas, A.; Gualtieri, M.; Saenko, K.; Platt, R. Grasp pose detection in point clouds. *Int. J. Robot. Res.* **2017**, *36*, 1455–1473. [\[CrossRef\]](#)
3. Chavan-Dafle, N.; Rodriguez, A. Stable prehensile pushing: In-hand manipulation with alternating sticking contacts. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–26 May 2018; pp. 254–261.
4. Hill, J. Real time control of a robot with a mobile camera. In Proceedings of the 9th International Symposium on Industrial Robots, Washington, DC, USA, 13–15 May 1979; pp. 233–246.
5. Su, J.; Ma, H.; Qiu, W.; Xi, Y. Task-Independent robotic uncalibrated hand-eye coordination based on the extended state observer. *IEEE Trans. Syst. Man Cybern. Part B (Cybernetics)* **2004**, *34*, 1917–1922. [\[CrossRef\]](#) [\[PubMed\]](#)
6. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [\[CrossRef\]](#)
7. Zhang, Y.; Qiu, Z.; Zhang, X. A Simultaneous Optimization Method of Calibration and Measurement for a Typical Hand-Eye Positioning System. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 1–11. [\[CrossRef\]](#)
8. Haixia, W.; Fan, X.; Lu, X. Application of a hand-eye self-calibration technique in robot vision. In Proceedings of the 25th Chinese Control and Decision Conference (CCDC 2013), Guiyang, China, 25–27 May 2013.
9. Hervé, J.-Y.R.; Sharma, R.; Cucka, P. Toward robust vision-based control: Hand/eye coordination without calibration. In Proceedings of the 1991 IEEE International Symposium on Intelligent Control, Arlington, VA, USA, 13–15 August 1991.
10. Su, J.; Qielu, P.; Yugen, X. Dynamic coordination of uncalibrated hand/eye robotic system based on neural network. *J. Syst. Eng. Electron.* **2001**, *12*, 45–50.
11. Zhenzhen, X.; Su, J.; Ma, Z. Uncalibrated hand-eye coordination based HRI on humanoid robot. In Proceedings of the 33rd Chinese Control Conference, Nanjing, China, 28–30 July 2014.
12. 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.* **2017**, *37*, 421–436. [\[CrossRef\]](#)
13. Cooperstock, J.R.; Milios, E.E. Self-supervised learning for docking and target reaching. *Robot. Auton. Syst.* **1993**, *11*, 243–260. [\[CrossRef\]](#)
14. 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, Australia, 21–26 May 2018.
15. 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.



16. 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, Australia, 21–26 May 2018; pp. 1–8.
17. Yu, K.T.; Rodriguez, A. Realtime state estimation with tactile and visual sensing. Application to planar manipulation. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–26 May 2018; pp. 7778–7785.
18. Vierendeck, 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.
19. Kalashnikov, D.; Irpan, A.; Pastor, P.; Ibarz, J.; Herzog, A.; Jang, E.; Quillen, D.; Holly, E.; Kalakrishnan, M.; Vanhoucke, V.; et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In Proceedings of the Conference on Robot Learning, Zürich, Switzerland, 29–31 October 2018; pp. 651–673.
20. Kehoe, B.; Matsukawa, A.; Candido, S.; Kuffner, J.; Goldberg, K. Cloud-Based robot grasping with the google object recognition engine. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013.
21. Wang, Z.; Xu, Y.; He, Q.; Fang, Z.; Xu, G.; Fu, J. Grasping pose estimation for SCARA robot based on deep learning of point cloud. *Int. J. Adv. Manuf. Technol.* **2020**, *108*, 1217–1231. [[CrossRef](#)]
22. Fang, H.-S.; Wang, C.; Gou, M.; Lu, C. Graspnet-1billion: A large-scale benchmark for general object grasping. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020.
23. Bai, Q.; Li, S.; Yang, J.; Song, Q.; Li, Z.; Zhang, X. Object Detection Recognition and Robot Grasping Based on Machine Learning: A Survey. *IEEE Access* **2020**, *8*, 181855–181879. [[CrossRef](#)]
24. Zhang, H.; Wang, F.; Wang, J.; Cui, B. Robot grasping method optimization using improved deep deterministic policy gradient algorithm of deep reinforcement learning. *Rev. Sci. Instrum.* **2021**, *92*, 025114. [[CrossRef](#)] [[PubMed](#)]
25. Mahler, J.; Matl, M.; Satish, V.; Danielczuk, M.; Deroose, B.; McKinley, S.; Goldberg, K. Learning ambidextrous robot grasping policies. *Sci. Robot.* **2019**, *4*. [[CrossRef](#)] [[PubMed](#)]
26. Marcos, A.; Izaguirre, A.; Graña, M. Current research trends in robot grasping and bin picking. In Proceedings of the 13th International Conference on Soft Computing Models in Industrial and Environmental Applications, San Sebastian, Spain, 6–8 June 2018; Springer: Cham, Switzerland, 2018.
27. Takuya, T.; Hashimoto, M. Model-Less estimation method for robot grasping parameters using 3D shape primitive approximation. In Proceedings of the 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), Munich, Germany, 20–24 August 2018.
28. Delowar, H.; Capi, G.; Jindai, M. Evolution of deep belief neural network parameters for robot object recognition and grasping. *Procedia Comput. Sci.* **2017**, *105*, 153–158.
29. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015.
30. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv* **2015**, arXiv:1506.01497. [[CrossRef](#)]
31. 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.
32. Sanderson, A. Image based visual servo control using relational graph error signal. In Proceedings of the International Conference on Cybernetics and Society, Tokyo, Japan, 3–7 November 1980.
33. Le, H.M.; Jiang, N.; Agarwal, A.; Dudík, M.; Yue, Y.; Daumé, H. Hierarchical imitation and reinforcement learning. *arXiv* **2018**, arXiv:1803.00590.
34. Kanbar, M.S. Tangram Game Assembly. U.S. Patent 4,298,200, 3 November 1981.
35. Gonzalez, R.C.; Woods, R.E.; Eddins, S.L. *Digital Image Processing Using MATLAB*; Pearson Education India: Chennai, India, 2004.
36. Nise, N.S. *Control Systems Engineering, (With CD)*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
37. Newell, A.; Simon, H.A. Computer science as empirical inquiry: Symbols and search. In *ACM Turing Award Lectures*; ACM: New York, NY, USA, 2007; p. 1975.
38. Luger, G.F. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*; Pearson Education: London, UK, 2005.
39. Fikes, R.E.; Nilsson, N.J. Strips: A new approach to the application of theorem proving to problem solving. *Artif. Intell.* **1971**, *2*, 189–208. [[CrossRef](#)]
40. Chowdhury, A.; Koval, D. *Fundamentals of Probability and Statistics*; CRC Press: Boca Raton, FL, USA, 2009.