

Article

Robot Intelligent Grasp of Unknown Objects Based on Multi-Sensor Information

Shan-Qian Ji, Ming-Bao Huang  and Han-Pang Huang *

Robotics Laboratory, Department of Mechanical Engineering, National Taiwan University, Taipei 10617, Taiwan; shanqianji@gmail.com (S.-Q.J.); d00522011@ntu.edu.tw (M.-B.H.)

* Correspondence: hanpang@ntu.edu.tw; Tel.: +886-2-3366-2700

Received: 4 March 2019; Accepted: 30 March 2019; Published: 2 April 2019



Abstract: Robots frequently need to work in human environments and handle many different types of objects. There are two problems that make this challenging for robots: human environments are typically cluttered, and the multi-finger robot hand needs to grasp and to lift objects without knowing their mass and damping properties. Therefore, this study combined vision and robot hand real-time grasp control action to achieve reliable and accurate object grasping in a cluttered scene. An efficient online algorithm for collision-free grasping pose generation according to a bounding box is proposed, and the grasp pose will be further checked for grasp quality. Finally, by fusing all available sensor data appropriately, an intelligent real-time grasp system was achieved that is reliable enough to handle various objects with unknown weights, friction, and stiffness. The robots used in this paper are the NTU 21-DOF five-finger robot hand and the NTU 6-DOF robot arm, which are both constructed by our Lab.

Keywords: contact modelling; force and tactile sensing; grasping and manipulation; grasp planning; object features recognition; robot hand-arm system; robot tactile systems; sensor fusion; slipping detection and avoidance; stiffness measurement

1. Introduction

Rapid technology development is enabling intelligent robots to be used in many fields, such as medicine, the military, agriculture, and industry. A robot's ability is a key function to grasp and manipulate an object that helps people with complicated tasks. In order to provide daily support by using humanoid hands and arms [1,2], robots must have the ability to grasp a variety of unseen objects in human environments [3].

In the unstructured environment, a common gripper has the limitation of not being able to grasp a great variety of objects. Therefore, studies need to be focused on using a multi-fingered robot hand to grasp objects with different shapes. Second, recognition and grasping of unknown objects in a cluttered scene have been very challenging to robots. This study attempted to develop a grasping system that is fast, robust and does not need a model of the object beforehand in order to reduce reliance on preprogrammed behaviors. The contributions of this work are summarized as follows:

- We present a method that combines three-dimensional (3D) vision and robot hand action to achieve reliable and accurate segmentation given unknown cluttered objects on a table.
- We propose an efficient algorithm for collision-free grasping pose generation; the grasp pose will be further checked for its grasp quality. Experiments show the efficiency and feasibility of our method.
- We addressed the problem associated with grasping unknown objects and how to set an appropriate grasp force. To fulfill this requirement, we adopted a multi-sensor approach to identify the stiffness of the object and detect slippage, which can promote a more lifelike functionality.

- By fusing all available sensor data appropriately, an intelligent grasp system was achieved that is reliable and able to handle various objects with unknown weights, friction, and stiffness.

2. Preliminary Knowledge

2.1. Point Cloud Processing

For a grasping task [4–7], finding the position and pose of the target object must be done first. The real environment (Figure 1a) is complex, and the raw point cloud is typically dense and noisy. This will hurt performance in object recognition (Figure 1b). Many point cloud processing methods can be used to deal with the point cloud set, and then useful information can be extracted to reconstruct the object.

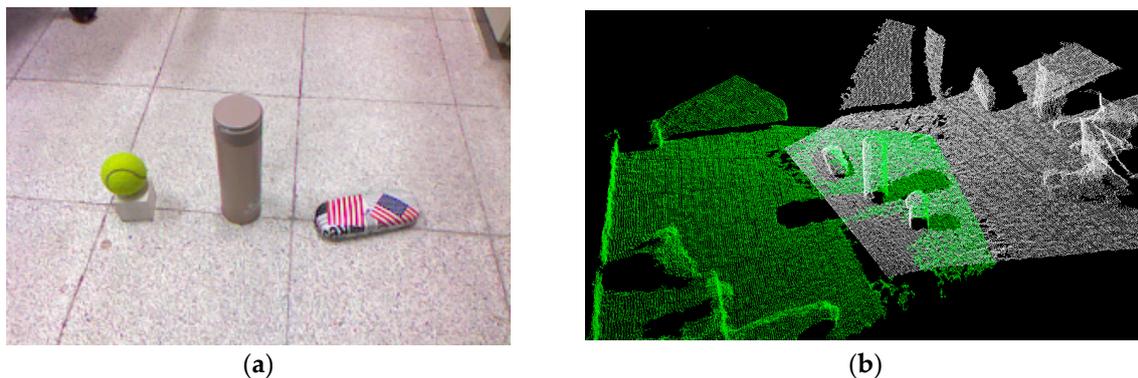


Figure 1. Real environment and point cloud: (a) Real environment; (b) Point cloud from the depth sensor.

2.1.1. Pass-Through Filter

The raw data of a point cloud contains numerous useless points. Therefore, a pass-through filter is used to set a range over the 3D space so that points within that range are kept unchanged, and points outside of the range are removed. In Figure 2, we show the point cloud after using a pass-through filter.

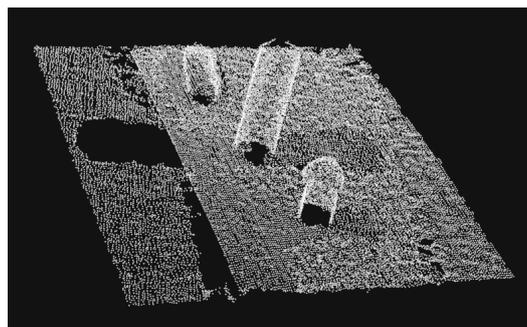


Figure 2. The output of pass-through filter.

2.1.2. Down-Sampling

A point cloud might contain points that provide no additional information due to noise or inaccuracy of the capture. On the other hand, when there are too many points in a point cloud, processing can be computationally expensive. The process of artificially decreasing the number of points in a point cloud is called down-sampling. In this study, the algorithm we used was voxel grid down-sampling. Figure 3 shows a point cloud before down-sampling and the same point cloud after down-sampling.

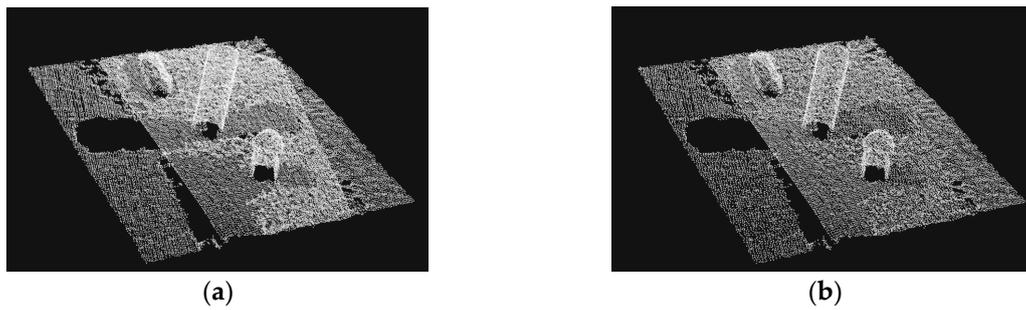


Figure 3. Down-sampling: (a) Original point cloud; (b) After down-sampling.

2.1.3. Random Sample Consensus (RANSAC)

First, we assumed objects are placed on flat surfaces. In a point cloud representation of a scene, a random sample consensus [8] can help to distinguish flat surfaces. Isolating a point cloud cluster that represents the unknown object is done by removing the entire points on the flat surface. After the planar model has been found, we can easily remove the ground and keep those objects on the ground. Figure 4 shows the objects retrieved from the original point cloud after image processing.

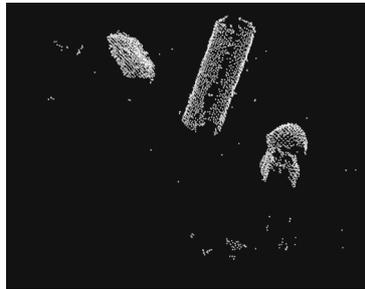


Figure 4. Objects after removing the ground.

2.1.4. Euclidean Segmentation

After separating the object from the table, the next problem was how to detect the number of objects remaining. The simple method we used was Euclidean segmentation. The result is shown in Figure 5, where we used different colors to represent different objects.

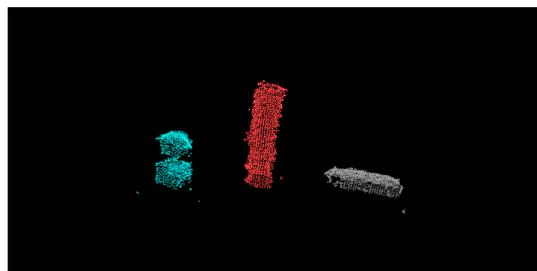


Figure 5. Point cloud after performing segmentation.

2.2. Oriented Bounding Box and Decomposition

For a common object, there are many possible grasps, and searching all possible grasps is difficult and unrealistic. In order to select a good grasp for an unknown object, we adopted an oriented bounding box (OBB) to identify the shape and pose of objects. The box is one of the most common shape primitives to represent unknown objects [6,9–11], and it can help us quickly find a suitable grasp. We assume that a hand can grasp an object if the bounding box of the object can be grasped. We created the OBB via principal component analysis (PCA). The steps involved in the OBB method are as follows:

- Compute the centroid (c_0, c_1, c_2) and the normalized covariance.
- Find the eigenvectors for the covariance matrix of the point cloud (i.e., PCA).
- These eigenvectors are used to transform the point cloud to the origin point such that the eigenvectors correspond to the principal axes of the space.
- Compute the max, min, and center of the diagonal.

Figure 6 shows a joystick, a glass case, and a tennis ball, each with their bounding box as generated using the algorithm discussed above.

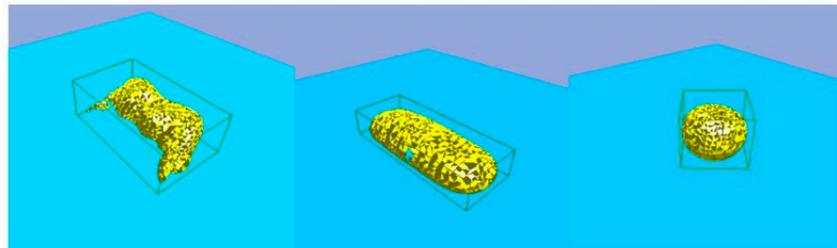


Figure 6. The bounding box of a set of objects.

In order to get better resolution of the actual models, we approximated the objects with bounding box decomposition. For this process, the number of boxes needed depends on the accuracy of approximation that is required. The principle of bounding box decomposition is that each bounding box is six-sided, and the opposing sides are parallel and symmetric. Considering the computation time, we determined that there were three valid split directions that were perpendicular to planes A, B, and C (Figure 7). The goal was to find the best split plane between each of these opposing sides.

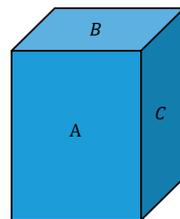


Figure 7. Parallel box cutting planes.

The split measure was defined as:

$$\alpha(d, x), \quad (1)$$

where d is one of the three split directions and x is the split position on this axis. For each x that cuts the target point cloud, the original point cloud can be divided into two subsets of data points. These can be used as inputs for the OBB algorithm to produce two child bounding boxes. Then we can obtain $\alpha(d, x)$ as the fraction between the sum volume of the two parts of the OBB and the whole volume. It is intuitive that the minimum is the best split (Figure 8). In this way, the box and the data points can be iteratively split, and the new boxes will better fit the shape.

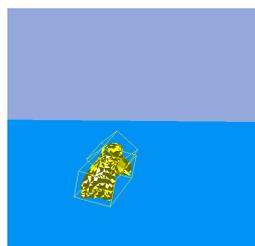


Figure 8. Best cuts.

Additionally, for the purpose of efficiency, an iterative breaking criterion was set as:

$$\Theta^* = \frac{V(C_1) + V(C_2)}{V(P)} \geq \theta, \quad (2)$$

where P is the current (parent) box, C_1 and C_2 are the two child boxes produced by the split, V is a volume function, and θ is the threshold value.

The bounding box decomposition algorithm has two constraints. First, if the Θ is too high, the split is not valuable. Second, boxes that include a very low number of points that can be considered noise must be removed.

2.3. Object Segmentation in a Cluttered Scene

When there are objects in a cluttered scene, it is difficult for a robot to distinguish among the unknown objects and grasp them separately. However, a robot's object segmentation ability is key when it must work in human environments. In our study, the goal was to distinguish the object states by using the robot visual system and inference of object logic states through taking specific actions. This can instruct a robot about an accurate segment object point cloud in the scene without supervision.

The framework of unknown object segmentation is shown in Figure 9. First, the vision module was used to capture the 3D point cloud of the current task environment, and the unknown object was segmented through Euclidean segmentation and saved as the original object dataset. At the same time, the current visual detection results were sent to the object logic state inference module to generate the initial logic state of each object. This was done because the spatial position relationship between the objects was unclear, such as with the presence of an occlusion and superposition. The initial object dataset is a generally inaccurate representation of the current environment.

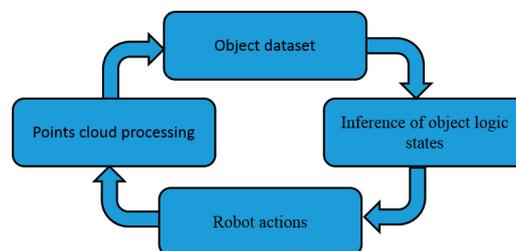


Figure 9. The framework of unknown object segmentation.

Specifically, when the robot performs a certain action, it feeds back the execution result of the action to the object logic state inference module, updates the logic state of the related object, and then updates the object point cloud. By observing the point clouds of objects that do not conform to the changed logic state, the object that is blocked or superimposed can be found, and the visual inspection result is fed to the object logic state inference module to update the object's logic state space.

First, define the object and logic state:

$$s_i = [Z_1, Z_2, Z_3], \quad (3)$$

where Z_1 is equal to 1 or 0, which indicates if object i is in or out of the robot's hand. Variable Z_2 presents the number of objects that are piled around object i . If Z_2 is equal to 0, it indicates that no objects are piled around object i . Variable Z_3 represents the number of objects under object i . If Z_3 is equal to -1 , it indicates that object i is in the air. If variable Z_3 is equal to 0, it indicates that object i is on the ground.

According to the definition of object logic state, the two most important robot actions must be the focus: `PickUp()` and `PutDown()`. The action `PickUp(object i)` can be used to grasp object i from the table, and `PutDown(object i, j)` can be used to load the grasped object i into the target position j . It is

assumed that only the robot's actions change the cluttered scene. The examples for inference of the object logic states based on action feedback are shown in Table 1.

Table 1. The rules for inference of object logic states.

Pre-States	Actions	Current States
$s_1 = \{0, 0, 0\}$	PickUp(object 1)	$s_1 = \{1, 0, -1\}$
$s_1 = \{0, 0, 0\}$	PutDown(object 2, object 1)	$s_1 = \{0, 2, 0\}$
$s_1 = \{1, 0, -1\}$	PutDown(object 1, object 2)	$s_1 = \{0, 0, 2\}$

Combining the above-mentioned logic state of objects and the basic actions of robots, the fundamental steps of object segmentation in a cluttered scene are shown in Figure 10. This method is demonstrated by the example shown in Figure 11.

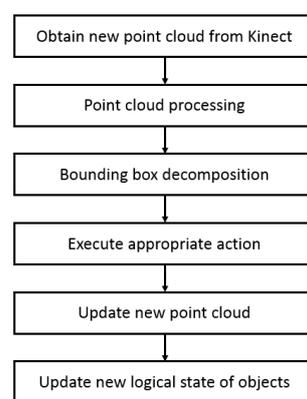


Figure 10. The flow of object segmentation.

According to the result of the bounding box decomposition (Figure 11a), box 1 and box 2 may be an integrated object or two separate objects. Thus, the reasonable action is picking up the top box.

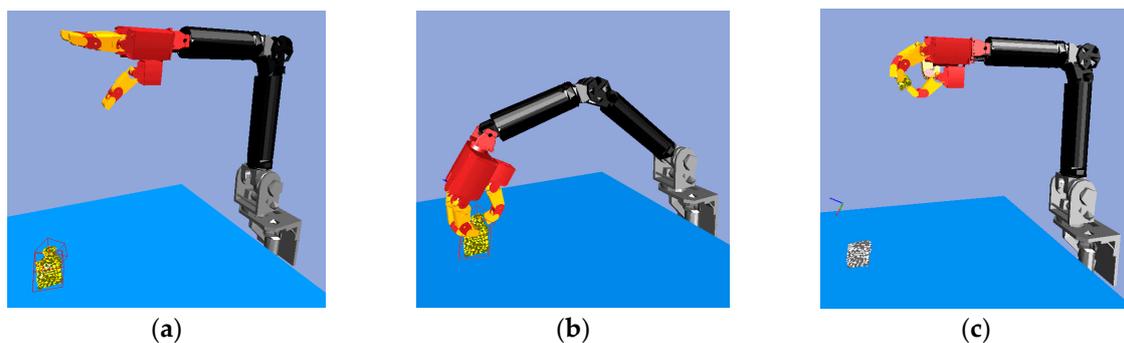


Figure 11. Object segmentation in the simulator: (a) Object's original position; (b) Previous moment; (c) Current moment.

After executing the PickUp() action (Figure 11b), it can be determined whether the object was blocked by observing the variation of the point cloud according to Table 2, where the first rule can be used to find an overlaid object. After object 1 was grasped, if it was an integrated object, the vision system should no longer detect the point cloud in the object's original position. If the point cloud is detected in that location, it indicates the point cloud is a new object 2, and that object 2 was overlaid with object 1 before the robot hand acted. The current pose of object 1 is determined based on the pose of the robot hand, and no visual algorithm is needed to re-detect it. The point cloud model of object 1 is updated according to the difference between the point cloud of object 1 at the previous moment and the point cloud of object 2 at the current moment, with a result shown in Figure 11c.

Table 2. The rules for segmentation point sets.

Action	Previous State	Current State	Detect Point Cloud in Previous Position		Detect Point Cloud in Current Position	
			Yes	No	Yes	No
PickUp(object 1)	$s_1 = \{0, 0, 0\}$	$s_1 = \{1, 0, -1\}$	covered object	normal	object 1	err
PickUp(object 1)	$s_1 = \{0, 0, 2\}$	$s_1 = \{1, 0, -1\}$	object 2	err	object 1	err
PutDown(ground)	$s_1 = \{1, 0, -1\}$	$s_1 = \{0, 0, 0\}$	err	normal	object 1	err
PutDown(object 1, object 2)	$s_1 = \{1, 0, -1\}$	$s_1 = \{0, 0, 2\}$	err	normal	object 1 and 2	err

3. Grasp Planning

In general, there are many pose candidates for grasping a target object. Additionally, contact points between the object and the robot need to check if it is a stable grasp, such as with the force closure property. In this section, we describe an efficient grasp planning algorithm for high-quality and collision-free grasping pose generation. The complete grasp planning approach has two main aspects: The first is checking that the grasping pose has a collision-free path for a 6-DOF arm using rapidly exploring random trees (RRTs). The next is fast grasping quality analysis in wrench space.

3.1. Grasp Strategy

Like human beings grasp things, robot grasping movements can be defined as using one of two reach-to-grasp movements: top grasp or side grasp [12], as shown in Figure 12.



Figure 12. The robot grasping movements: (a) Top grasp; (b) Side grasp.

Humans may grasp contact points that are on the longer side of an object. Therefore, we decide the grasping direction according to the height of the bounding box of the target object. If the height is adequate, we can use a side grasp movement. Otherwise, we use a top grasp movement.

Thus, based on the OBB of the object, we can sample the position and orientation of the end-effector of the robot arm, as shown in Figure 13. Then, inverse kinematics (IK) are solved for the robot arm and a check is done for whether any collisions may occur between the robot arm system and the environment. If a collision occurs, then the grasping pose is resampled in other directions until the pose can avoid the collision. Next, we find a better grasp pose by looking at two criteria. One is the grasp pose can find a short collision-free path or the pose results in a high-quality grasp.

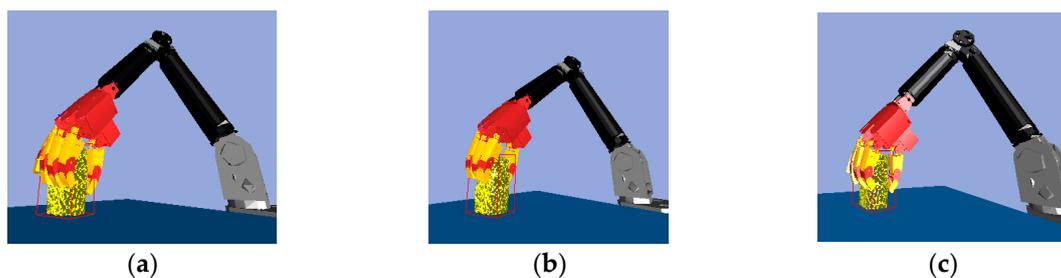


Figure 13. The sampling of the robot arm: (a) Top grasp movement; (b) A short collision-free path; (c) High-quality grasp.

3.2. Path Planning

Path planning involves generally searching the path in a configuration space (C -space). For the robot arm, C means the workspace of the robot arm and each $q \in C$ is a joint angle. Previous research has focused on only one goal for the purposes of motion path planning. However, there are multiple possible goals in the real world. The multi-objective RRT-Connect path planner was designed specifically for a real robot made as a high degrees of freedom (DOFs) kinematic chain [13].

The idea of multi-objective is illustrated in Figure 14. First, we set some goals in the configuration space where it is possible that some objectives are invalid, such as goal 3. Therefore, we need to check whether or not the goals are feasible according to the RRT-Connect algorithm. Let T_{init} and T_{goal} be the tree generated by q_{init} and $\{q_{goal}\}$, respectively. The multi-objective RRT-Connect algorithm is given in Table 3.

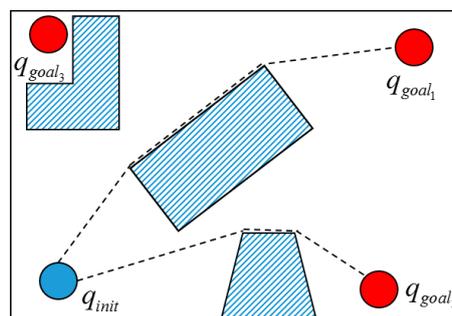


Figure 14. Multi-goal planner.

Table 3. Multi-objective RRT-connect algorithm.

-
1. Input $k = 1, r = 1, m, \{q_{init}\}$ and $\{q_{goal_1}, \dots, q_{goal_n}\}$;
 2. $q_{rand} \leftarrow randomConfig()$;
 3. Find the nearest neighbor n_{init} of q_{rand} to T_{init} ;
 4. Define k ;
 5. Make a segment toward q_{rand} from n_{init} with a fixed incremental distance d ;
 6. Define $q_{newinit}$;
 7. If the segment connecting n_{init} and $q_{newinit}$ is collision free then
 8. extend T_{init} (generate by $\{q_{newinit}, T_{init}\}$);
 9. Find the nearest neighbor n_{goal} of q_{rand} to T_{goal} ;
 10. Make a segment toward q_{rand} from n_{goal} with fixed incremental distance d ;
 11. Define $q_{newgoal}$;
 12. If the segment connecting n_{goal} and $q_{newgoal}$ is collision free then
 13. extend T_{goal} (generate by $\{q_{newgoal}, T_{goal}\}$);
 14. If T_{init} and T_{goal} are connected then
 15. Output the path, and end.
 16. Else replace k by $k + 1$, and return to Step 4
 17. Else if $r < m$ then replace r by $r + 1$, and return to Step 2
 18. Else end;
 19. Else if $r < m$ then replace r by $r + 1$, and return to Step 2
 20. Else end;
-

The random Config() function generates random $q_{rand} \in C$. The extend function T_{init} is illustrated in Figure 15. Each iteration attempts to extend the RRT by adding a new vertex that is biased by a randomly selected configuration. First, we find the vertex q_v that is the nearest vertex to q in tree T . Then we extend the tree toward q from q_{near} with ad-hoc variable distance d and assign a new variable $q_{newinit}$ before the collision event is checked.

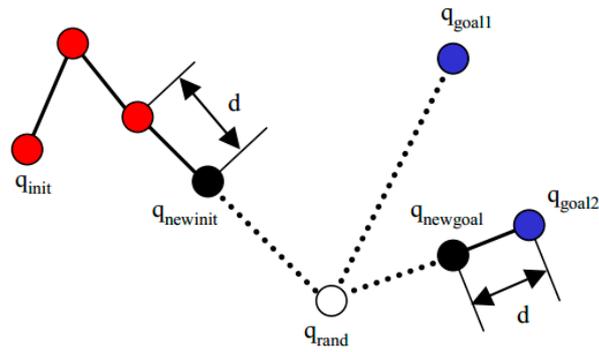


Figure 15. The extend function.

Consider the robot arm joint space path planning. Figure 16 shows that with the same start configuration but different goal configurations, there will be a different path. The yellow object is the target object, and the white object is the obstacle which the robot arm and hand must avoid. The multi-objective RRT-Connect algorithm is used to search the six-dimensional C-space of the robot arm in order to find the collision-free path. Figure 16a shows the grasping pose that had a short collision-free path.

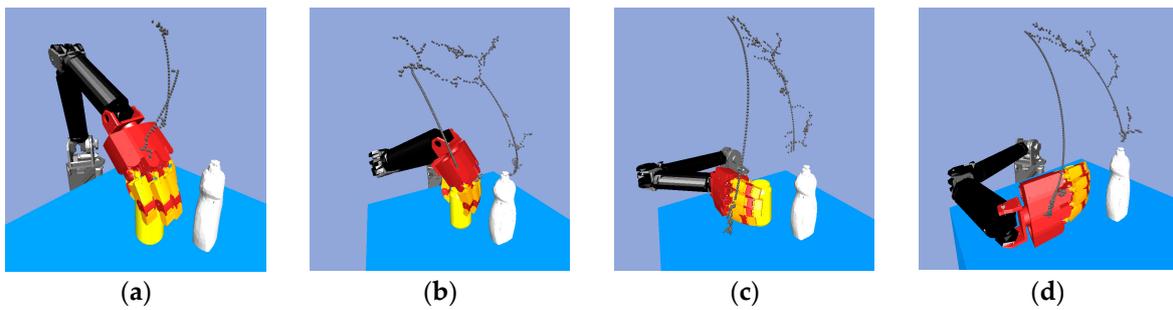


Figure 16. Multi-objective RRT-Connect in the simulator: (a) Path size: 59; (b) Path size: 238; (c) Path size: 191; (d) Path size: 231.

After searching for a path, we then needed to flatten, or prune, the path by removing any unnecessary waypoints, after which we can obtain a suitable trajectory for real robot grasping and manipulation. There are some efficient path pruning algorithms, like that shown in Ref. [14]. Even though the pruned path is not optimal, this method can remove most of the unnecessary waypoints very quickly, eliminating the chattering phenomenon in the RRT-Connect algorithm. Figure 17 shows the path before using the path pruning algorithm and the path after running the path pruning algorithm.



Figure 17. Path pruning algorithm: (a) Original path; (b) The path through path pruning algorithm.

3.3. Grasp Analysis

After path planning is complete, the robot arm moves into position with a given orientation, and the robot hand performs continuous grasping until it touches the object. Here, the robot hand was treated as a simple gripper to reduce its number of DOFs, but a multi-fingered robot hand can adapt to the object better than a simple gripper because it has five fingers that can be independently adjusted to the object's geometry. When all fingers are in contact with the object's surface, it is still not clear whether or not the robot hand can firmly grasp the target object. Thus, grasp analysis is necessary. Grasp analysis is a matter of determining, given an object and a set of contacts, whether the grasp is stable using common closure properties.

A grasp is commonly defined as a set of contacts on the surface of an object [15–17]. The quality measure is an index that quantifies the effectiveness of a grasp. Force closure is a necessary property for grasping that requires a grasp to be capable of resisting any external wrench on the object, maintaining it in mechanical equilibrium [16].

One important grasp quality measure [18] often used in optimal grasp planning [19] assesses the force efficiency of a grasp by computing the minimum of the largest wrench that a grasp can resist, over all wrench directions, with limited contact forces [20], which equals the minimum distance from the origin of the wrench space to the boundary of a grasp wrench set.

The friction cone given by use of an equation is a convex cone, which consists of all non-negative combinations of the primitive contact force set U_i , defined as follows:

$$U_i = \left\{ f_i | f_{n_i} = 1, \sqrt{f_{t_i}^2 + f_{o_i}^2} = \mu \right\}, \quad (4)$$

The set U_i is the boundary of the friction cone. The image of U_i in the wrench space \mathbb{R}^6 through the contact map G_i is called the primitive contact wrench set, which is expressed as:

$$W_i = G_i(U_i), \quad (5)$$

Let W_{L_1} be the union of W_i and W_{L_∞} be the union of the Minkowski sum for any choice of them:

$$W_{L_1} = \bigcup_{i=1}^m W_i, \quad W_{L_\infty} = \bigoplus_{i=1}^m W_i, \quad (6)$$

The grasp wrench set is defined to be the convex hull of W_{L_1} or W_{L_∞} , which is denoted by $W_{L_1}^{co}$ or $W_{L_\infty}^{co}$:

$$W_{L_1}^{co} = \left\{ \sum_{i=1}^m G_i f_i | f_i \in \text{friction cone}, \sum_{i=1}^m f_{n_i} = 1 \right\}, \quad (7)$$

$$W_{L_\infty}^{co} = \left\{ \sum_{i=1}^m G_i f_i | f_i \in \text{friction cone}, f_{n_i} \leq 1 \right\}, \quad (8)$$

Because the calculation of the Minkowski sum is more difficult, we chose the Equation (9) to compute the grasp wrench space (GWS). We can use some properties of GWS to evaluate the grasp quality. If the interior of $W_{L_1}^{co}$ is nonempty and contains the origin of the wrench space \mathbb{R}^6 , the grasp has force closure and is stable. Furthermore, the minimum of the largest resultant wrenches that a grasp can resist, over all wrench directions, with limited contact forces is an important quality measure of a force closure grasp. It can be formulated as the minimum distance between the origin of the wrench space \mathbb{R}^6 and the boundary of $W_{L_1}^{co}$, and the magnitude of the quality measure Q is defined as:

$$Q = \min_{v \in W_{L_1}^{co}} \|v\|, \quad (9)$$

where v is \mathbb{R}^6 vector. Figure 18 shows an example of the quality measure in the simulator. The negative score in Figure 18 means that the zero set $\{0\}$ was not contained in the GWS. In other words, the grasp was unstable. For grasps with a positive score, the higher the score, the better the grasp's capability of resisting a perturbation. Usually, Q ranges from 0.01 to 0.30 for stable grasping. We considered that grasps of more than 0.01 were sufficiently stable.

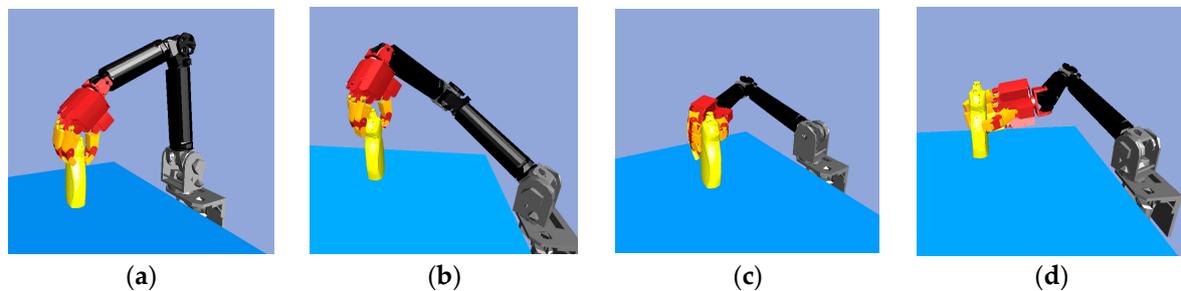


Figure 18. Quality measure example for a bottle: (a) Path size: 0.0135; (b) Path size: 0.001; (c) Path size: 0.017; (d) Path size: -0.27 .

4. Real-Time Grip Force Selection and Control

The previous sections illustrated our method of estimating an object's shape and determining a robust and safe grasping configuration. However, while executing the grasping motion, the appropriate force required to immobilize an object still needs to be set. The force regulation is also very important for a precision grasp, especially when grasping is followed by manipulation.

This paper presents a methodology for choosing an appropriate grasping force when a multi-finger robot hand grasps and lifts an object without knowing its weight, coefficient of static friction, or stiffness. To fulfill these requirements, a method that measures object stiffness and detects slippage by using a multi-sensor approach is proposed. The desired grasping force can be set with an upper limit to avoid damage and a lower limit to avoid slippage according to the object's measured characteristics. Figure 19 shows the concept of a grip force selection structure.

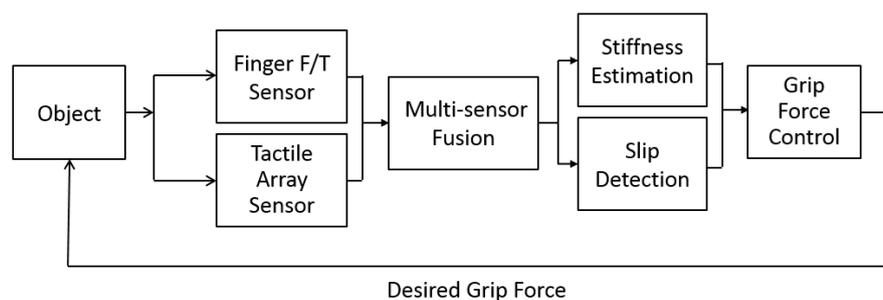


Figure 19. Grip force selection structure.

The control flowchart shown in Figure 20 consists of two stages, the pre-grasp stage and the grasping stage. In the pre-grasp stage, the robot hand is positioned, and using the position controller, the robot hand fingers are closed on the target object until the normal force feedback is above a certain threshold. The threshold is chosen according to the stiffness measurement using a very small value in order to avoid damaging the object. Once all the fingers are in contact with the object, the position controller is stopped, and the hand enters the grasping stage. In the grasping stage, the robot hand will be switched to position-based force control.

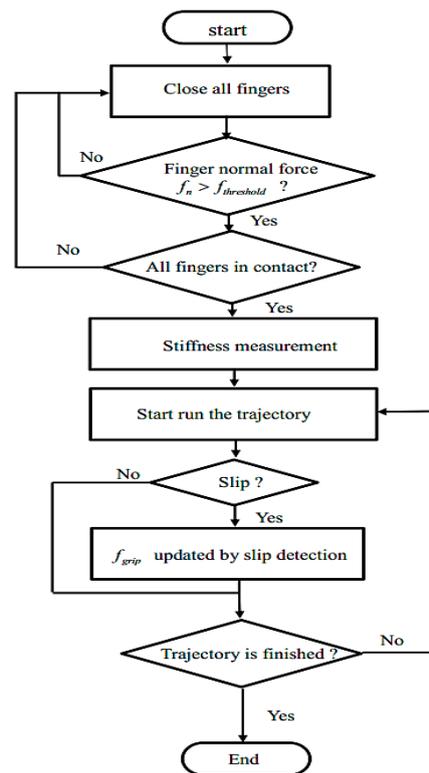


Figure 20. Control diagram of the grip controller.

4.1. Stiffness Measurement

When the robot hand is to grasp an object, the stiffness of the target object is a very important parameter that must be determined. Objects in our study were therefore divided into three categories according to their stiffness: a rigid object, a soft object, and a very soft object. Each category was defined by the upper limit grasp force.

A rigid object has high stiffness, and will not be deformed by the grasping force, such as a glass cup or metal bottle. For this category of objects, the upper limit force is defined as the maximum limit of the applied force of the robot finger.

Soft objects like plastic or paper cups have medium level stiffness and can be deformed when the robot hand increases the grasping force. For this category, the force that deforms objects is greater than the lift-off grasping force, so the appropriate upper limit force is set to have a minimal difference between the two forces.

The third category contains very soft objects such as sponges and other objects with little stiffness. This kind of object is greatly deformed by a slight grasping force. However, this deformation differs from soft objects because, in this case, it is acceptable for the robot hand to keep a stable grasp even if it causes a slight deformation of the object surface.

The concept of our method is that objects with different stiffness will have different mechanical responses to the initial contact. Thus, the force data to determine the stiffness is collected and processed. More details about our method will be discussed in the next subsection.

The stiffness measurement is taken using the following steps. First, the robot closes its fingers until it detects initial contact with the object. Once the hand contacts with the surface of the object, it continues to close for a fixed distance along the normal direction of the finger surfaces. Then the force curve is analyzed and features are selected to determine the object's stiffness. Both three-axis force/torque (F/T) sensor [21] and tactile sensor arrays [5] are employed to collect the force data, and the obtained fusion results are shown in Figure 21. As seen from the force curve, different stiffness will have different force curves. The harder the object, the faster the force increases and the greater the

steady force value when the loading is finished. What's more, there are transient and steady states encountered during the stiffness measurement process. For the transient state, the hand makes contact with the object at a constant velocity. The results indicate that the force is proportional to contact speed for harder objects. Thus, the force curve slope U_d was extracted as a feature to estimate the stiffness of a grasped object as:

$$U_d = \frac{1}{N_1} \sum_{k=i}^{k=N_1+i} F_k, \quad (10)$$

where i is the time at which the hand starts to close, and N_1 is the sampling number of the transient state. F_k is the normal force in the contact point at time k . For steady state, the object has been grasped. Further, the gripping force and the reverse force due to deformation have been balanced. According to Hooke's law, $F = KX$, the steady pressure value is proportional to the stiffness of the object. Thus, the final steady pressure value of the curve can be used as a feature to estimate the stiffness of a grasped object:

$$U_s = \frac{1}{N_2} \sum_{k=j}^{k=N_2+j} F_k, \quad (11)$$

where j is the time at which the hand stops, and N_2 is the sampling number of the steady state. Based on the above analysis, the characteristics of force during contact and steady-state force characteristics at equilibrium can be obtained.

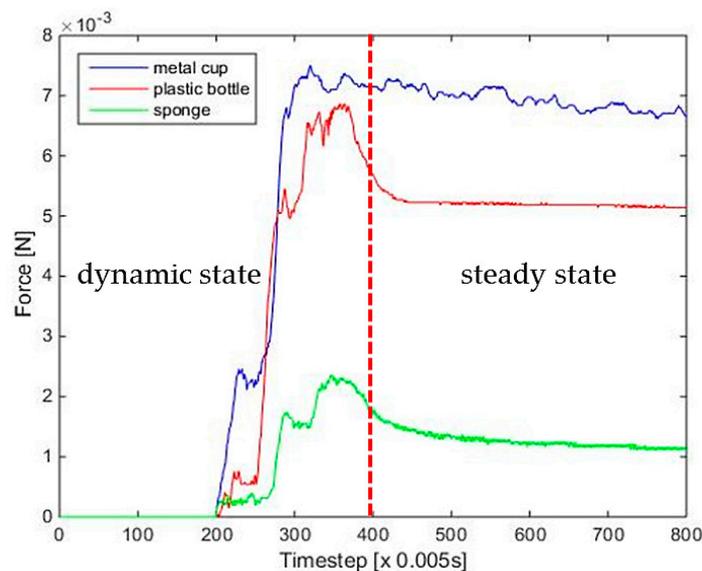


Figure 21. Force curves of different categories of objects.

After selecting the appropriate features, the k-nearest neighbors (K-NN) algorithm [22,23], was used to fuse these features and obtain more accurate stiffness recognition results. The K-NN algorithm is used to classify sample points into several distinct classes, and is summarized below:

- A positive integer k is specified along with a new sample.
- The k entries in the database that are closest to the new sample are selected.
- The most common classification of these entries is determined.
- This is the classification given to the new sample.

4.2. Slipping Detection and Avoidance

Robust slip detection ability [24] is one of the most important features needed for a grasping task. Knowledge about slip can help the robot prevent the object from falling down its hand. Furthermore, the sensation of slip is critical for a robot to grasp an object with minimal force [25].

In this study, we combined tactile sensor arrays [5] and three-axis force/torque (F/T) sensor [21] to obtain the slip signals. The framework of slip signal detection is shown in Figure 22. The slip feature based on frequency analysis and the slip feature based on motion estimation will be fused by support vector machine (SVM) algorithm.

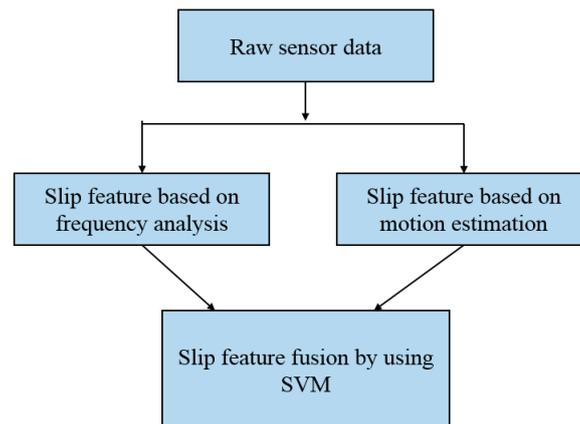


Figure 22. Slip detection and avoidance framework.

A number of studies have developed a method that observes the frequency content of the contact forces when slippage occurs [26,27]. When the relative speed of the fingers and objects are relatively low, the theory of friction and vibration shows that the slippage is an intermittent vibration that causes vibration of the gripping force. Slippage can, therefore, be detected by the high-frequency signal it generates. Thus, a wavelet transform [28,29], was used to analyze and process the finger three-axis force/torque (F/T) sensor information and extract the real-time slip feature. Considering the complexity and real-time performance of the algorithm, the simplest Haar wavelet in wavelet transforms was adapted to analyze the sensor information. The Haar scaling function is defined as:

$$\phi(x) = \begin{cases} 1, & \text{if } 0 \leq x < 1 \\ 0, & \text{elsewhere} \end{cases}, \quad (12)$$

The Haar wavelet function looks like:

$$\psi(x) = \phi(2x) - \phi(2x - 1), \quad (13)$$

The sensor force data $f(t)$ can be represented as the linear sum of the Haar scaling function and the Haar wavelet using Haar decomposition as:

$$f_j(t) = w_{j-1} + f_{j-1}(t), \quad (14)$$

where:

$$w_{j-1} = \sum_{k \in Z} b_k^{j-1} \psi(2^{j-1}t - k), \quad (15)$$

$$f_{j-1} = \sum_{k \in Z} a_k^{j-1} \phi(2^{j-1}t - k), \quad (16)$$

$$b_k^{j-1} = \frac{a_{2k}^j - a_{2k+1}^j}{2}, a_k^{j-1} = \frac{a_{2k}^j + a_{2k+1}^j}{2}, \tag{17}$$

and a_k^{j-1} are called the detail coefficient. The Haar wavelet is employed to transform the contact force information into the slip process and to extract the slip signal by using the detail coefficient. It is known from the definition of the detail coefficient that it can be expressed by the difference of the force signals of the adjacent two moments.

Vibration-based methods can suffer from increased sensitivity to robot motion [30]. A very simple alternative is the tracking of the center of mass (the intensity weighted centroid) [31]. But this method is quite sensitive to noise and saturation effects. In addition, a shifting of weight is interpreted as a translation even if the pressure profile is not moving at all. Thus, we considered using methods based on the convolution calculation [32].

The overall workflow is shown in Figure 23. Let the pixel intensity function of a sensor matrix T_n at time step n be denoted as $t_n(x, y)$, $T_n \in \mathbb{R}^{M \times N}$. The two-dimensional (2D) convolution of the discrete matrices, also known as complex-conjugate multiplication, is defined as:

$$t_{n-1}(x, y) * t_n(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} t_{n-1}(y, j) t_n(x - j, y - j), \tag{18}$$

This results in the convolution matrix C of size $(2M - 1) \times (2N - 1)$. The elements of C are therefore a measure of similarity between the two images, and the translation of the peak from the origin indicates the shift between them. In the context of slip detection, this relationship can be interpreted as a slip vector between two similar tactile sensor arrays profiles.

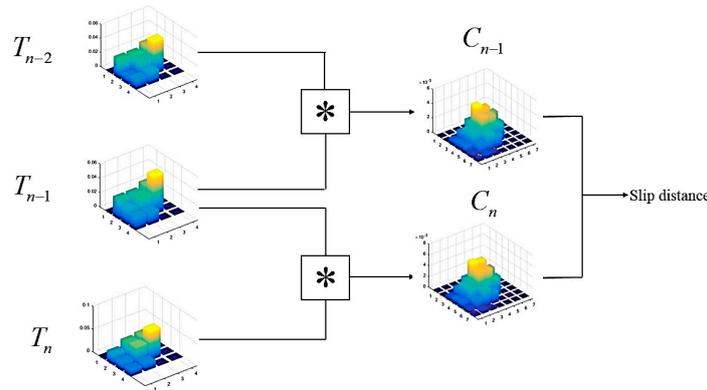


Figure 23. Slip detector based on motion estimation workflow.

The decision is made to implement a slip detection algorithm by Alcazar and Barajas [33]. First, two index matrices A and B of the same size as the convolution matrix and consisting of repeating rows and columns, respectively, are defined as:

$$A = \begin{bmatrix} 1 & 2 & \dots & 2N - 1 \\ 1 & 2 & \dots & 2N - 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \dots & 2N - 1 \end{bmatrix} - NB = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 2 & 2 & \dots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ 2M - 1 & 2M - 1 & \dots & 2M - 1 \end{bmatrix} - M, \tag{19}$$

In the slip detection loop, the convolution matrix C_{n-1} of the first pair of tactile matrices T_{n-2} and T_{n-1} is computed. A slip index along the X direction at time T is defined by:

$$\Delta x_{n-1} = E\left(\frac{A \cdot \mu_c^T}{sum(\mu_c)}\right), \tag{20}$$

Similarly, a slip index along the Y direction at time T is computed by:

$$\Delta y_{n-1} = E\left(\frac{\mu_r^T \cdot B}{\text{sum}(\mu_r)}\right), \quad (21)$$

where μ_c is a row vector containing the mean value of each column from the convolution matrix C_{n-1} . In contrast, μ_r is a column vector containing the mean value of each row from the convolution matrix C_{n-1} . $E()$ and $\text{sum}()$ denote the mean value and the sum of the elements of the vector, respectively. At the next time, the previous step was repeated. Again, the column and row means of defining the resulting convolution matrix C_n and the displacements were computed. The final slip signal was computed with:

$$X_n = \sqrt{(\Delta x_n - \Delta x_{n-1})^2 + (\Delta y_n - \Delta y_{n-1})^2}, \quad (22)$$

In this study, we fused the two slip signals mentioned above along with the SVM to improve the identification accuracy of slip detection performance. The SVM is a machine learning method based on statistical learning theory (SLT) [34] that minimizes empirical risk and can, therefore, solve linear and nonlinear classification problems.

4.3. Real-Time Grasping Control

The purpose of the grasping control is to adjust the grasping force to prevent slippage and limit deformation of unknown objects. The controller must have a slippage adaptive function to adjust the grasping force in real time.

Let F_{des} be the desired grip force (normal force) in each fingertip in a no-slip situation. Let F_{grip} be the updated desired grip force that accounts for any slip conditions. Define F_{grip} to be:

$$F_{grip}[k+1] = F_{grip}[k] + \alpha\beta, \quad (23)$$

where α is a gain factor that is derived according to the object's stiffness. A high stiffness object will have a bigger α , and this can help the hand quickly eliminate slippage. A low stiffness object will have a smaller α in order to avoid distortion of the object by increasing the grip force too quickly. In this paper, we have defined three levels of rigid categories, so there will be three corresponding α values. β is the slip signal derived from SVM results.

When slippage is detected, the regulated force must be applied almost instantaneously to ensure that the robot hand acts immediately. If the force is applied too slowly, the object will continue to slip and may even fall. Thus, a proportional-derivative (PD) position controller was used to achieve stable grasping control as follows:

$$U = \left\{ \begin{array}{ll} K_p(F_{grip} - F) + K_d\left(\frac{dF}{dt}\right), & \text{if } F \leq F_{\text{limit}} \\ 0, & \text{if } F > F_{\text{limit}} \end{array} \right\}, \quad (24)$$

where U is the output of position controller that determines each fingertip's normal direction in Cartesian space, K_p and K_d are the proportional and derivative parameters, and F is the real feedback force from the sensors in the fingertips. F_{limit} is set according to the object's stiffness.

5. Experiment Results

This section describes how the NTU 6-DOF robot arm [5,6,35–37] and the NTU five-finger robot hand were equipped with additional hardware and software to enable the resultant grasp of unknown objects. We combined the robot arm and robot hand to facilitate planning and real-time control, as shown in Figure 24.

This paper presents the on-going research of the National Taiwan University (NTU) five-finger robot hand. The purpose of the development of the NTU five-finger robot hand is for delicate dynamic grasp. To design a motor-driven, dexterous robot hand, we analyzed the human hand. Our design features a customized fingertip three-axis force/torque (F/T) sensor and joint torque sensors integrated into each finger, along with powerful super-flat brushless DC (BLDC) motors (The MAXON (Sachseln, Switzerland) motor measures only 23 mm in the outer diameter and creates a unit that is only 18.5 mm in height, with a motor weight of just 15 g. The rated speed and torque of the motor idle at 4140 rpm; the 3W motors are available in a 9V version and provide a maximum torque of 8.04 mNm. The motors include digital Hall sensors and digital magnetic Encoders. A linear per position resolution of 0.001 mm is obtained by coupling a MAXON Encoder MR (Type M 512 cpt) to the motor. and tiny harmonic drivers (HD) (Harmonic Drive™ gear AG, Limburg an der Lahn, Germany, HDUC-5-100-BLR, gear ratio 1:100). By using a steel coupling mechanism, the phalanx distal's transmission ratio is exactly 1:1 in the whole movement range. The rest of this paper presents a control strategy for the NTU five-finger robot hand. For robust grasp, we implemented classical impedance control. Our goal is to develop 21 DOFs dexterous robot hand. The hand has an independent palm and five identical modular fingers; each finger has three DOFs and four joints. We designed this robot hand by improving previously developed robot hand designs and by analyzing actual human hand motions. This NTU motor-driven, the five-finger robot hand can be equipped with a three-axis force/torque sensor [21] at each fingertip, as well as with developed distributed tactile sensor arrays with 376 detecting points on its surface [5]. The hand can communicate with external sources using technologies such as EtherCAT and controller area networks (CAN bus). To evaluate the performance of this robot hand design, we analyzed workspace, intersection volume, and manipulability, as shown in Figure 24.

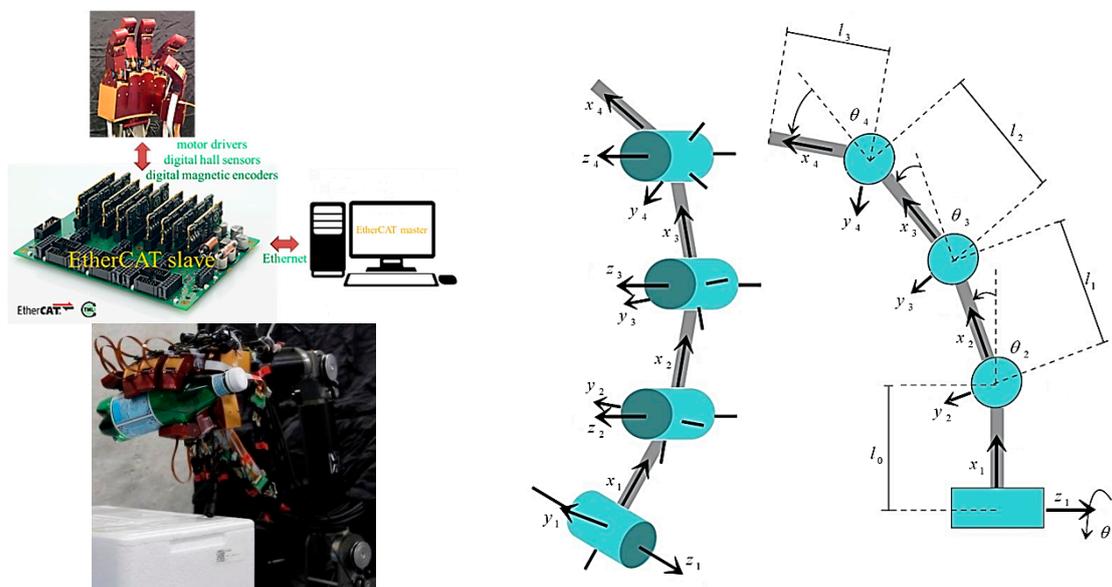


Figure 24. NTU five-finger hand with modular fingers (Four link manipulator to model one finger of the robot hand).

A dexterous robot hand needs at a minimum a set of force and position sensors to enable control schemes like position control and impedance control in autonomous operation and teleoperation. The aim of the sensory design is to integrate into the artificial hand a great number of different sensors in order to confer to the hand functionalities similar to that of a human hand. Sensor equipment for the NTU five-finger robot hand is shown in Table 4.

Table 4. Sensor equipment of one finger.

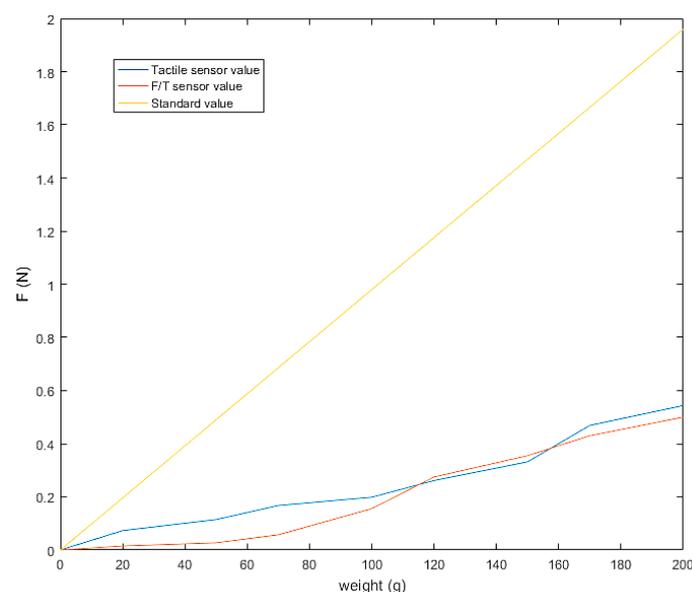
Sensor Type	Count/Finger
joint torque {current (torque) control loop in BLDC motor}	3
joint position	3
motor speed	3
distributed tactile sensor arrays (376 detecting points) (Tekscan, Inc., South Boston, MA, USA)	1
three-axis force/torque (each fingertip)	1
six-axis force/torque (the wrist joint) (Mini 40, ATI Industrial Automation, Apex, NC, USA)	1

The maximum payload of the NTU 6-DOF robot arm is over two kg. The specification of the NTU 6-DOF robot arm is shown in Table 5.

Table 5. Specification of NTU 6-DOF robot arm.

Total Weight	About 5.2 kg (exclude the shoulder)
Max. Payload	Over 2 kg
Max. Joint Speed	Exceed 110 deg./sec.
Max. Reachable distance	510 mm
Max. width	110 mm
Motor	Brushed DC X6
Reduction device	Harmonic drive
Transmission	Timing Belt, Gear

Because there are two kinds of the sensor on the fingertip, a three-axis F/T sensor and tactile sensor arrays, we calibrate and fuse the two sensors' normal force. We use weight groups (20, 50, 70, 100, 120, 150, 170 and 200 g) as the standard value and record the error between the measured value of the two sensors' normal force and weight's standard value. The result is shown in Figure 25. From the figure, in the range of 0 to 100 g, the linearity of the two sensors is different, while in the range of 100 to 200 g, the measured values of the two sensors are quite similar.

**Figure 25.** Force sensor calibration.

Therefore, we fuse the sensor data in the two areas separately using the least squares method. The result is shown in Figure 26. Let f_1 and f_2 denote three-axis F/T sensor and tactile sensor arrays measurements of normal force, respectively. In the 0–0.98 N range, the fusion force is $F = 1.8413f_1 + 3.6176f_2 - 0.0245$. In the 0.98–1.96 N range, the fusion force is $F = 4.9991f_1 - 1.2253f_2 + 0.1103$. The results show that the fused sensor values have good linearity with the standard value in the whole range.

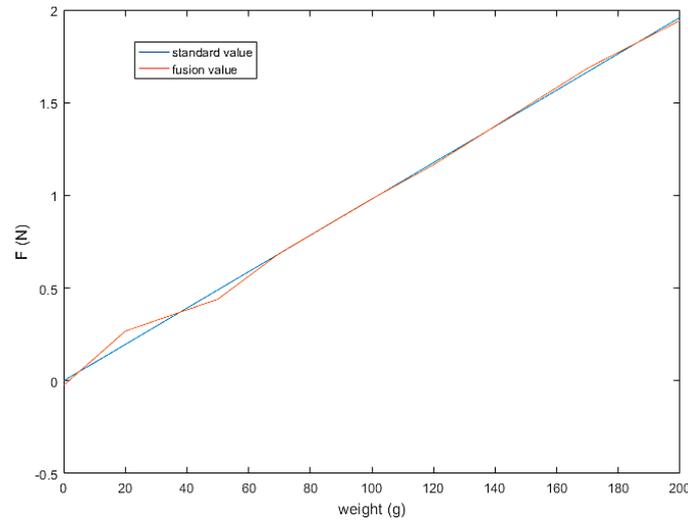


Figure 26. Sensor fusion result.

5.1. Experiment 1: Slipping Detection and Avoidance

This experiment was mainly aimed at verifying the accuracy of slip detection and the quick response of dynamic adjustment to the grasping force.

In order to collect the sensor data during the slip, we set up the experiment, as shown in Figure 27. First, we give the fixed initial grasp pose and grip force. When an initial steady grip was reached, a heavy load was added to the object (rice in the cup), as shown in Figure 28a. As the weight increased, the object slipped (Figure 28b), so the additional weight was used as a disturbance that caused relative slippage. Finally, the object slipped from the hand (Figure 28c), and we recorded the three-axis force/torque (F/T) sensor and tactile sensor arrays data.



Figure 27. Experiment setup for slip detection.

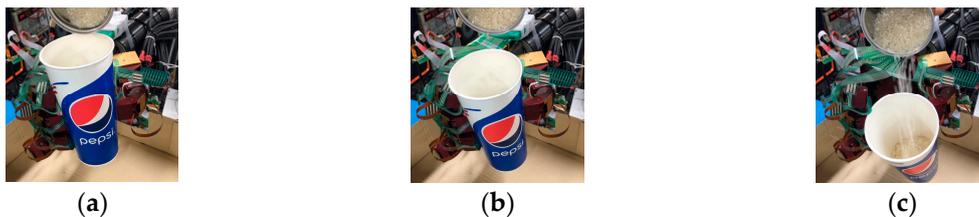


Figure 28. The process of slipping: (a) The heavy load was added to the object (rice in the cup); (b) Weight increased; (c) The object slipped from the hand.

The tangential force collected by the three-axis force/torque (F/T) sensor in the thumb is shown in the upper graph of Figure 29. From this figure, we can see that the raw data contained high-frequency random noise. In order to remove the noise, we used a low-pass filter to process the raw data. The result after filtering is shown in the bottom graph of Figure 29. The upper graph in Figure 30 presents the discrete wavelet transform from the finger three-axis force/torque (F/T) sensor. The bottom graph shows the displacement value from the tactile sensor arrays data. When $t = 1.28$ s, slippage occurred.

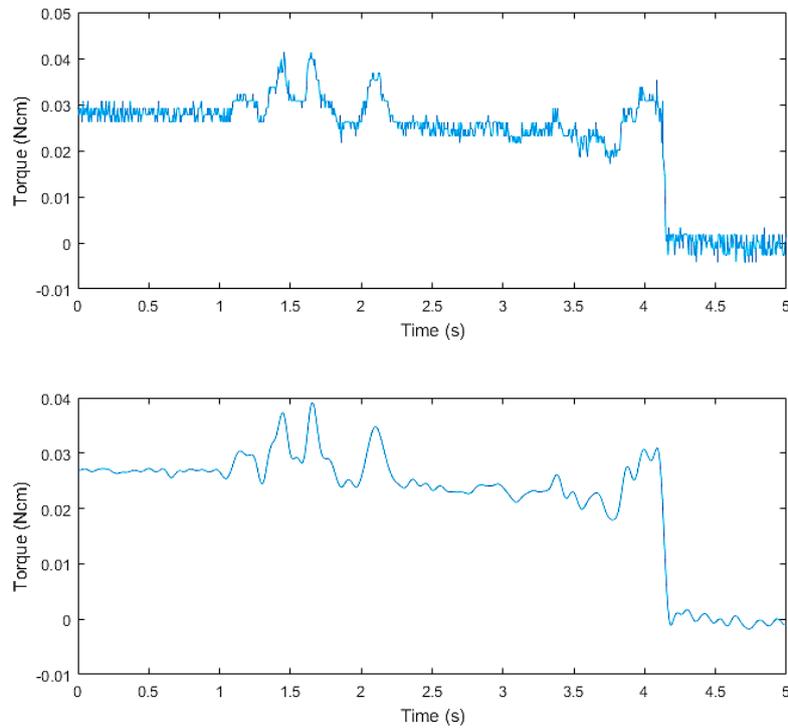


Figure 29. The tangential force of finger three-axis force/torque (F/T) sensor.

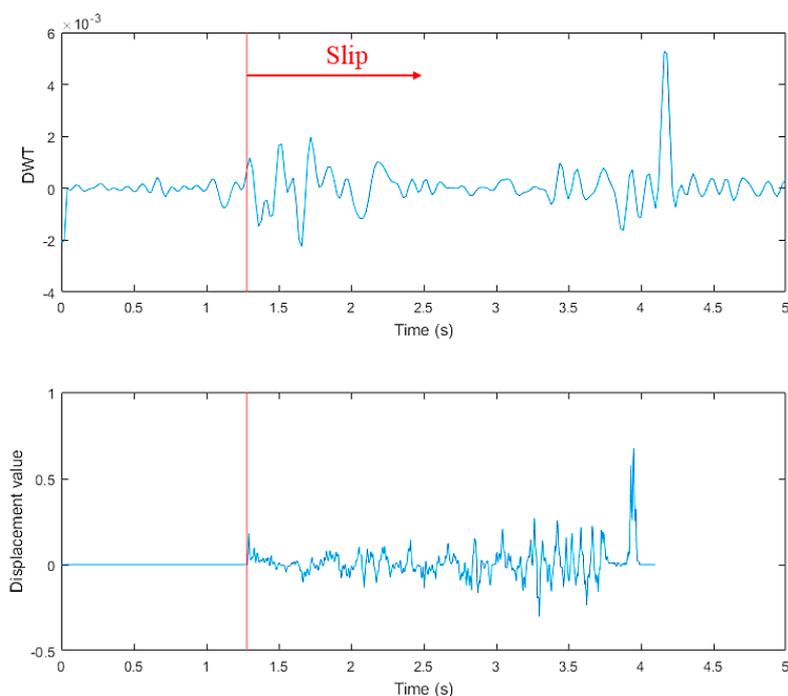


Figure 30. Results of slippage detection experiment.

The slip experiment dataset is shown in Figure 31, where the red spots represent the slip situation, and the blue spots represent the no-slip situation. Then we used LIBSVM [38] to train the model. In this study, we used a grid search method to find the best parameter (c , g) for an RBF kernel. As shown in Figure 32, we found that the best parameter was (8, 32), with a cross-validation rate of 97.5%.

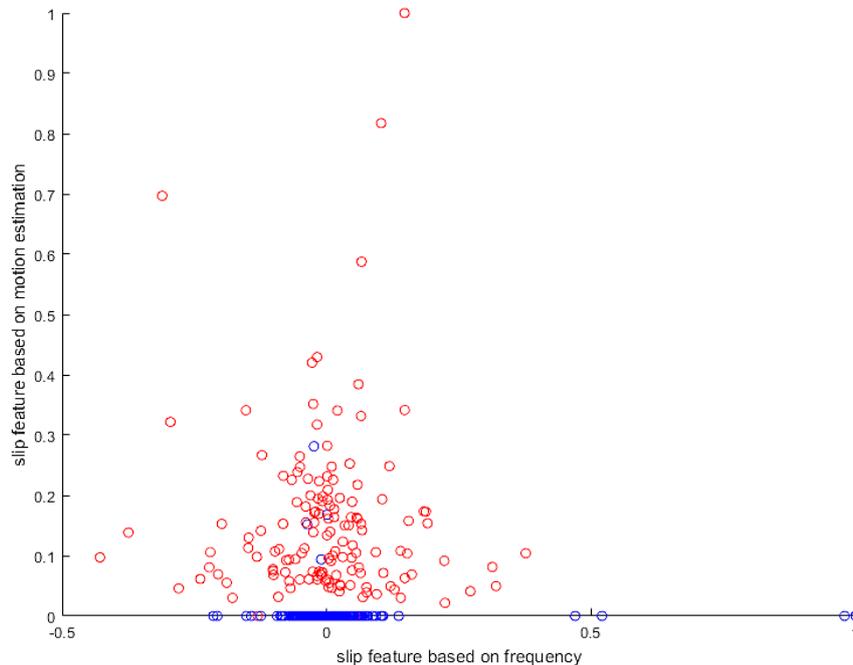


Figure 31. Dataset.

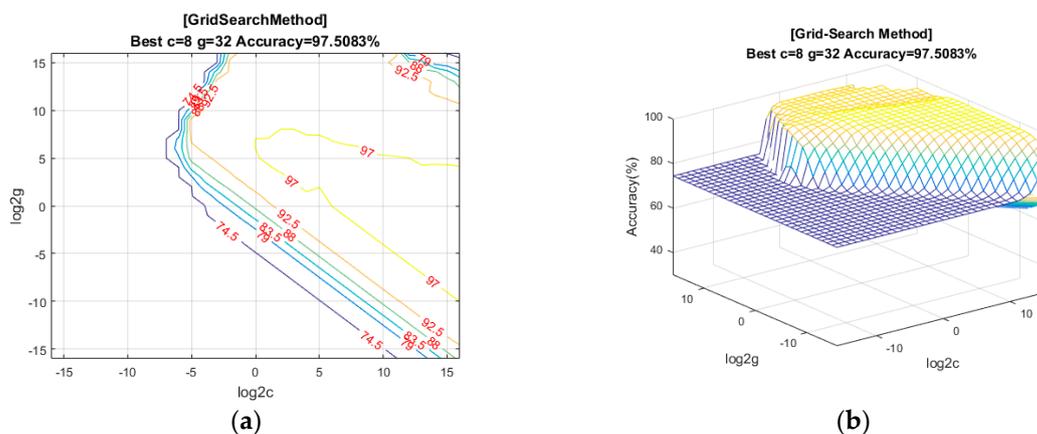


Figure 32. The result of the grid search method: (a) Grid-Search Method; (b) The cross-validation rate.

Next, we used the SVM predict function to help the robot hand detect the slip signal in real time. Similar to the previous experiment, the task of this experiment was to grasp the plastic cup with three fingers and gradually add rice to the cup. This time, we used slip detection to adjust the grip force in time to suppress the slip. The experimental process is shown in Figure 33. In this experiment, when an initial steady grip was reached, a heavy load was added to the object (rice in the cup) (Figure 33a–c). The additional weight was used as a disturbance that caused relative slippage, we succeeded in avoiding the slippage (Figure 33d–f). However, because we had not measured the stiffness of the object, when the robot hand came into contact with the object (heavy load), it caused deformation of the plastic cup (Figure 33g–i). The slip signal and the grip force change are shown in Figures 34 and 35.

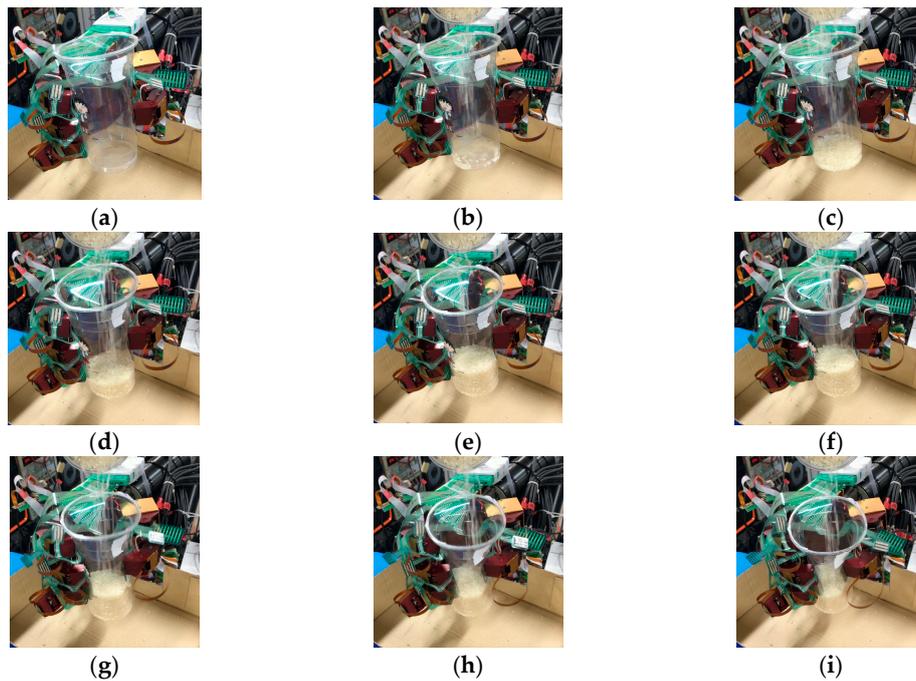


Figure 33. Slip prevention experiment: (a–c) When an initial steady grip was reached, a heavy load was added to the object (rice in the cup); (d–f) The additional weight was used as a disturbance that caused relative slippage, we succeeded in avoiding the slippage; (g–i) Finally, when the robot hand came into contact with the object (heavy load), it caused deformation of the plastic cup.

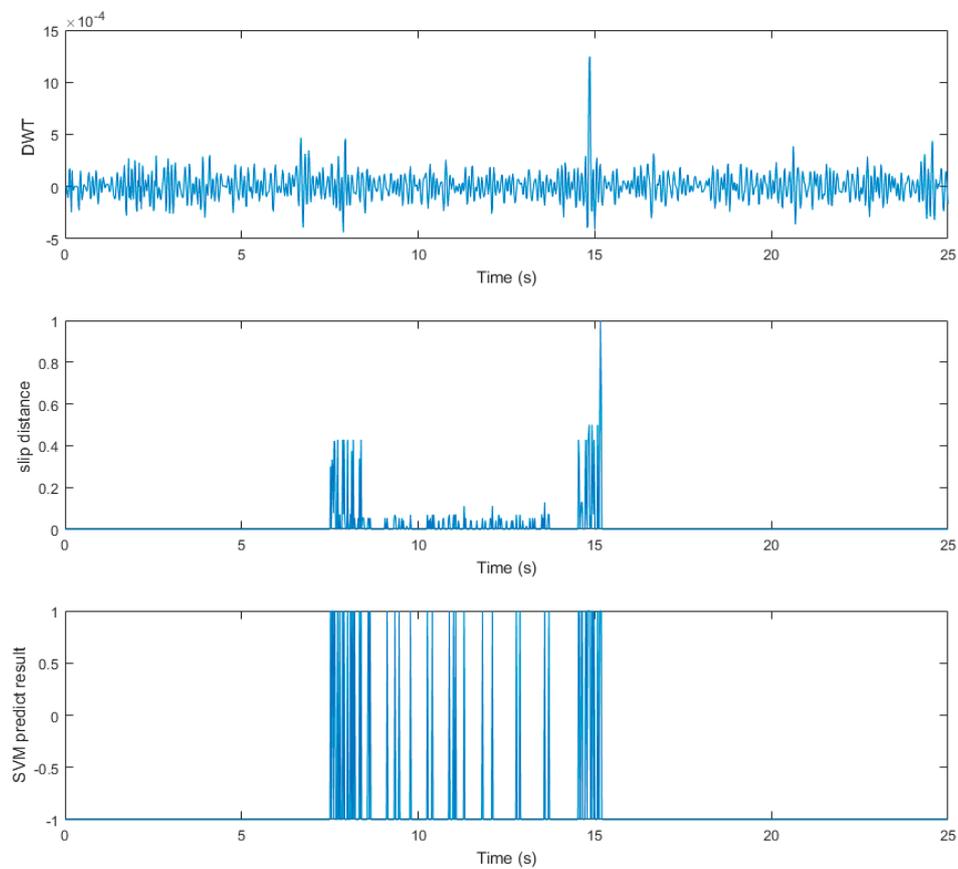


Figure 34. Slip signal.

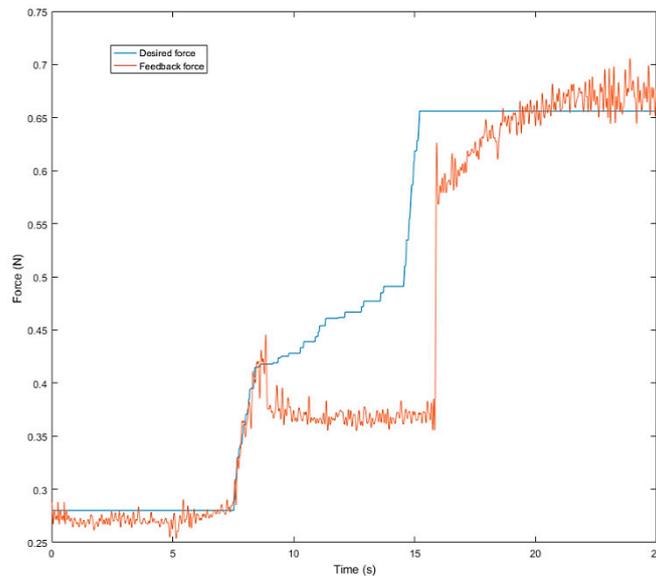


Figure 35. Grip force.

5.2. Experiment 2: Grasping in a Cluttered Scene

This experiment demonstrated grasping in a cluttered scene. The experimental environment used is shown in Figure 36. The objects were set on the blue table, and the robot hand-arm system was located at one end of the table. Two depth sensors were placed on one end of the table and on the side of the robot hand-arm system to capture the point cloud of the environment.



Figure 36. The experimental environment: (a) Real environment; (b) Objects in the simulator.

Because the NTU robot hand is a right hand, we predefined our task as being to clear the table from the right side to the left side. The first step was to grasp the milk tea. In the robot simulator block, the simulator chose the target object (milk tea). Then, the grasp strategy was used to find the grasp configuration of the robot hand-arm. The path planning used the RRT-Connect algorithm to find collision-free paths for grasping, as shown in Figure 37b.

Once the hand contacted the object, it first took a rigid measurement, as shown in Figure 38a. According to the result of K-NN (Figure 38b), the first object belonged to the low stiffness category. The planner generated a collision-free path from the current position to the target box area (Figure 37c), and then, in the grasping stage, the robot hand turned to position-based force control. According to the multi-sensor feedback as well as the slip detection and stiffness measurement, the robot hand could grasp the object successfully and place it in a specific area according to the object's stiffness. The procedure used for the real robot hand-arm system is shown in Figure 39.

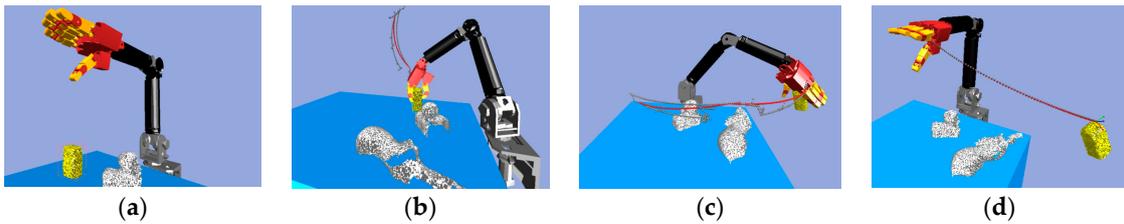


Figure 37. Experiment 2 in a simulator: (a) The target object (milk tea); (b) Collision-free paths for grasping; (c) RRT-Connect algorithm; (d) Successfully separated and placed in different boxes.

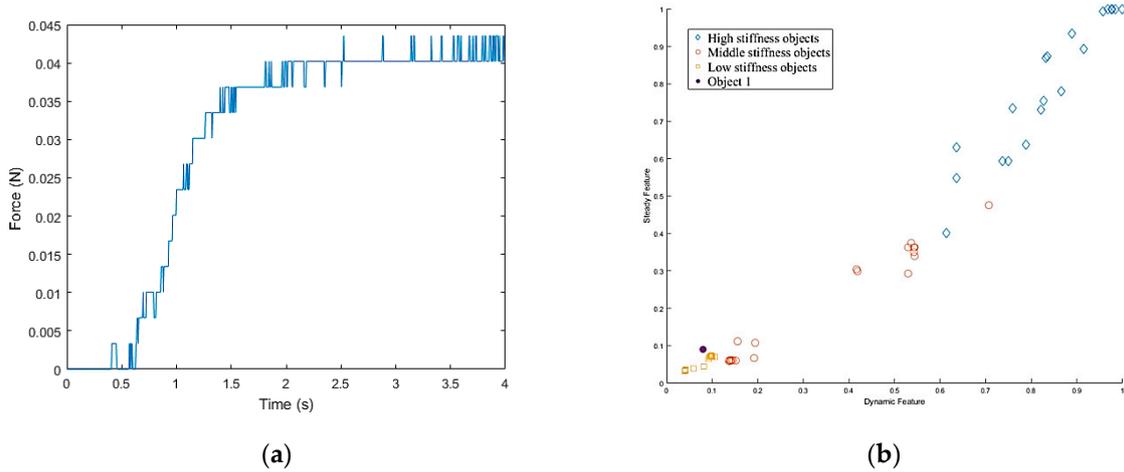


Figure 38. Object 1 stiffness measurement: (a) Stiffness; (b) K-NN.

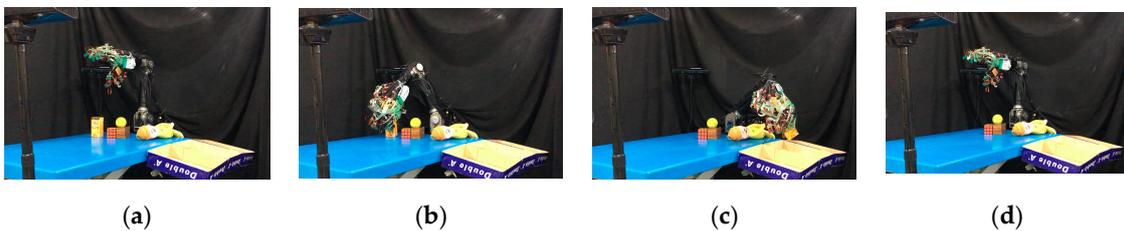


Figure 39. A procedure of grasping the first object: (a) The target object (milk tea); (b) Collision-free paths for grasping; (c) RRT-Connect algorithm; (d) Successfully separated and placed in different boxes.

Next, we found that in the middle of the table, there were three objects stacked together, but the vision system considered them to be one object only. As shown in Figure 40a, the target object was approximated by using a few bounding boxes, and the robot hand is first considered grasping the top box. The grasp poses are shown in Figure 40b. Similar to grasping object 1, the hand determined the stiffness measurement (Figure 41a), and K-NN indicated that object 2 belonged to the medium stiffness category. The procedures for the simulation and the real robot hand-arm are shown in Figures 40 and 42, respectively.

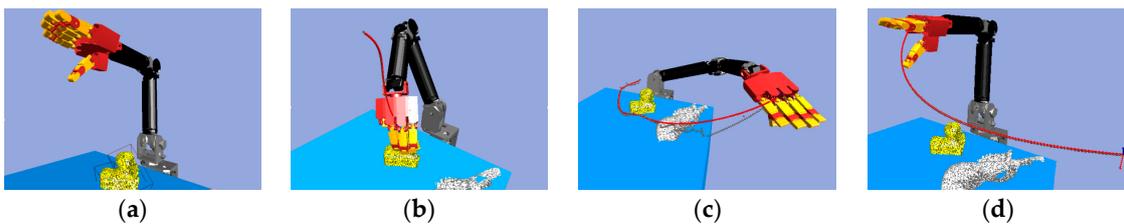


Figure 40. A procedure of grasping the second object in the simulator: (a) The target object (top box); (b) Collision-free paths for grasping; (c) RRT-Connect algorithm; (d) Successfully separated and placed in different boxes.

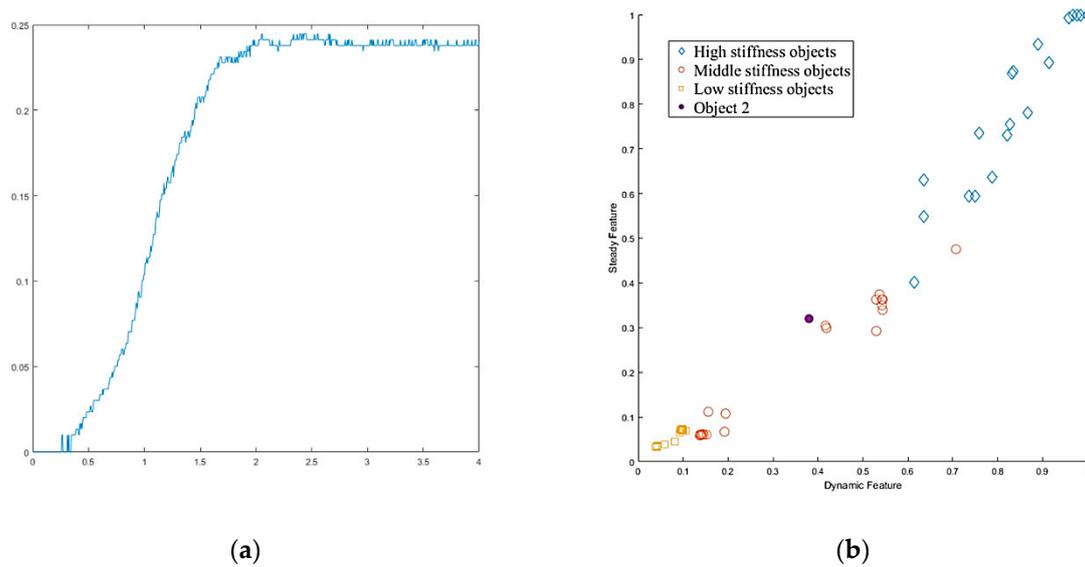


Figure 41. Object 2 stiffness measurement: (a) Stiffness; (b) K-NN.

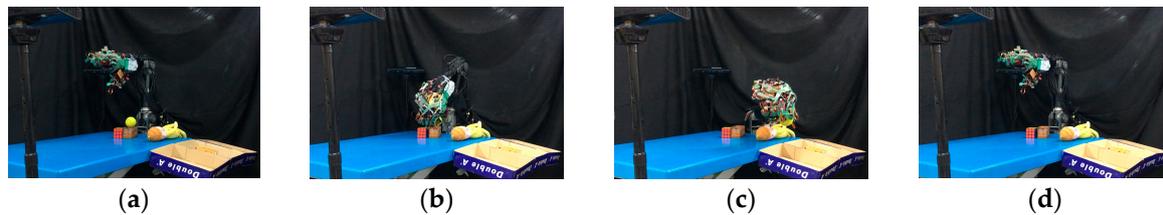


Figure 42. A procedure of grasping second object: (a) The target object (top box); (b) Collision-free paths for grasping; (c) RRT-Connect algorithm; (d) Successfully separated and placed in different boxes.

The vision system collected the point cloud in the scene again (Figure 43a) to update the object state according to the object state inference. After grasping object 2, there was still a point cloud in the same location, meaning there were multiple objects stacked in the same place, and the robot hand considered the next object remaining on the table to be object 3. Grasp planning was performed in a similar manner as the previous objects. The stiffness measurement (Figure 44a) and K-NN result (Figure 44b) indicated that object 3 belonged to the category of high stiffness. The procedures for the simulation and the real robot hand-arm are shown in Figures 43 and 45, respectively. Finally, the last object in the original location was considered to be object 4 is shown in Figure 46. The stiffness measurement (Figure 47a) and K-NN result (Figure 47b) indicated that object 4 also belonged to the high stiffness category. As the robot interacted with them, all the real objects were gradually separated. The procedures for the real robot hand-arm system and the simulation are shown from Figures 46–48.

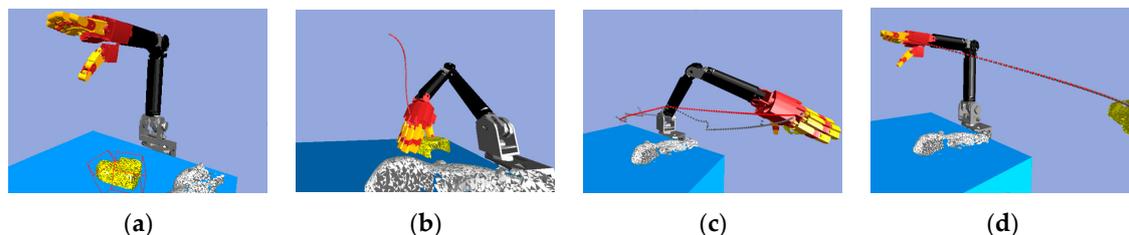


Figure 43. A procedure of grasping the third object in the simulator: (a) The target object; (b) Collision-free paths for grasping; (c) RRT-Connect algorithm; (d) Successfully separated and placed in different boxes.

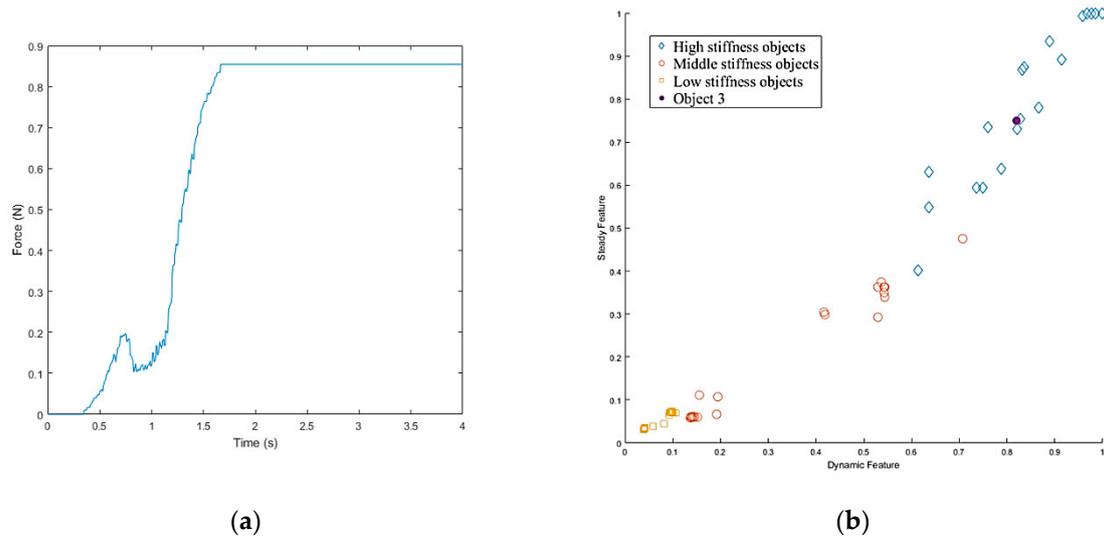


Figure 44. Object 3 stiffness measurement: (a) Stiffness; (b) K-NN.

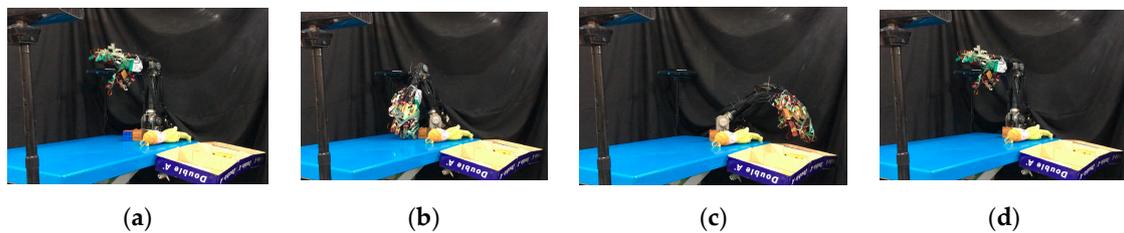


Figure 45. A procedure of grasping the third object: (a) The target object; (b) Collision-free paths for grasping; (c) RRT-Connect algorithm; (d) Successfully separated and placed in different boxes.

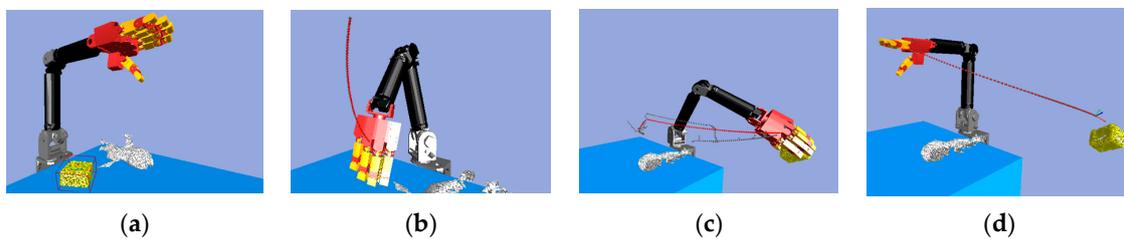


Figure 46. A procedure of grasping the fourth object in the simulator: (a) The target object; (b) Collision-free paths for grasping; (c) RRT-Connect algorithm; (d) Successfully separated and placed in different boxes.

The last object on the table, in a different location, was a cloth puppet, for which the bounding box is shown in Figure 49a. Based on this bounding box, the grasp planner chose an appropriate grasp configuration. According to the result of K-NN (Figure 50b), object 5 belonged to the low stiffness category. The procedures for the real robot hand-arm system and the simulation are shown in Figures 49 and 51, respectively. The classification results are shown in Figure 52.

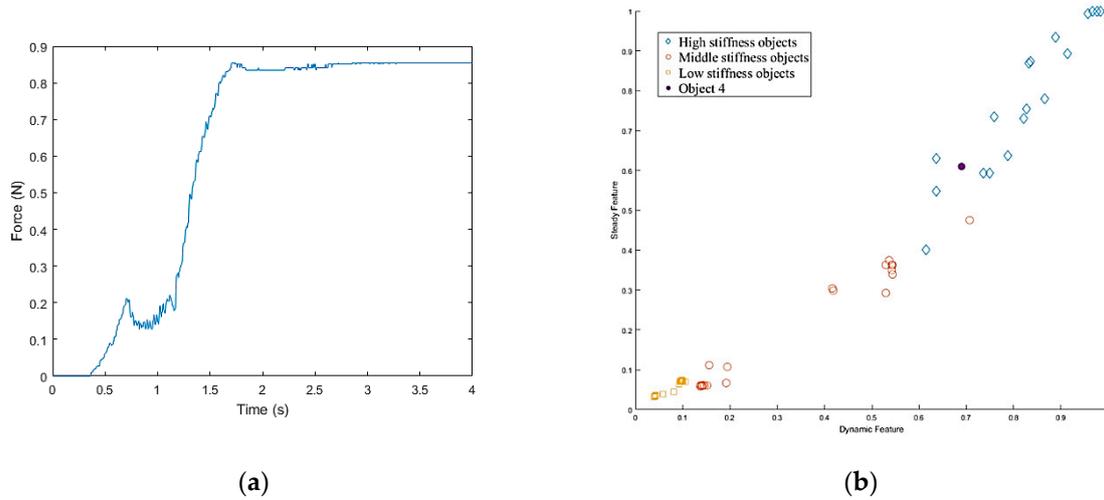


Figure 47. Object 4 stiffness measurement: (a) Stiffness; (b) K-NN.

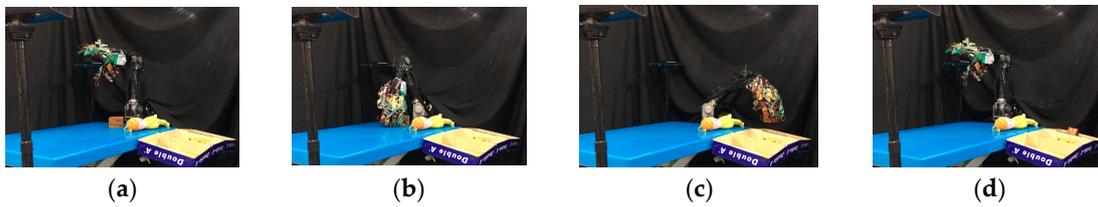


Figure 48. A procedure of grasping the fourth object: (a) The target object; (b) Collision-free paths for grasping; (c) RRT-Connect algorithm; (d) Successfully separated and placed in different boxes.

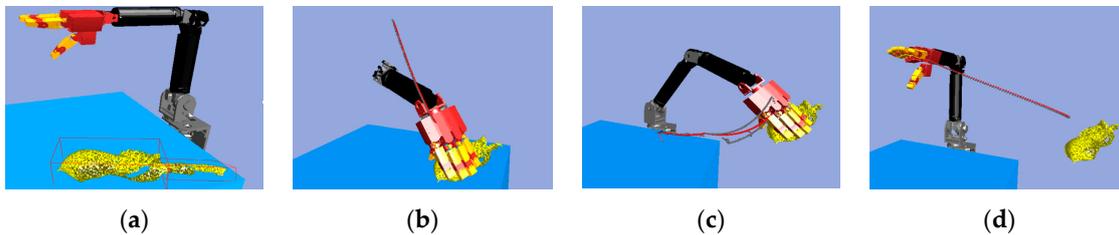


Figure 49. A procedure of grasping the fifth object in the simulator: (a) The target object (low stiffness category); (b) Collision-free paths for grasping; (c) RRT-Connect algorithm; (d) Successfully separated and placed in different boxes.

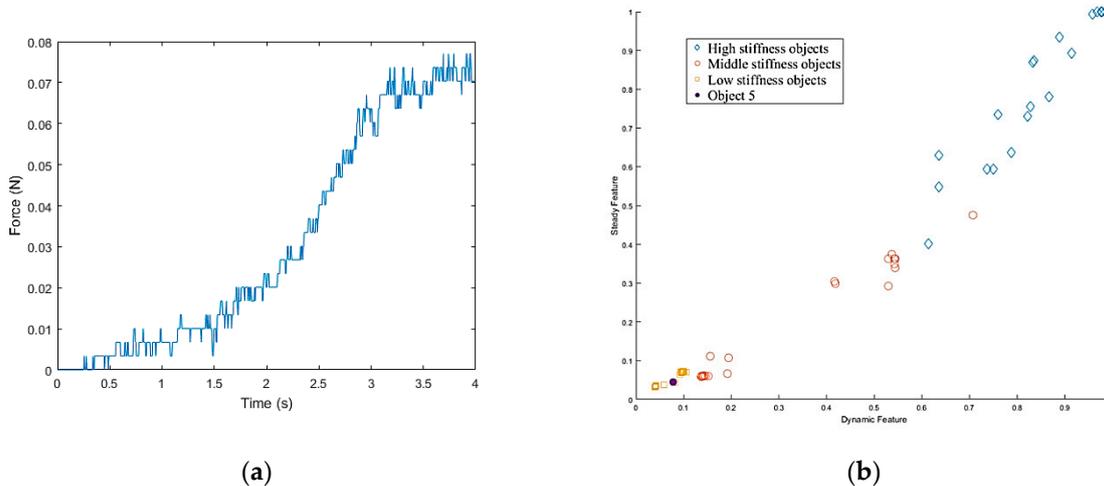


Figure 50. Object 5 stiffness measurement: (a) Stiffness; (b) K-NN.

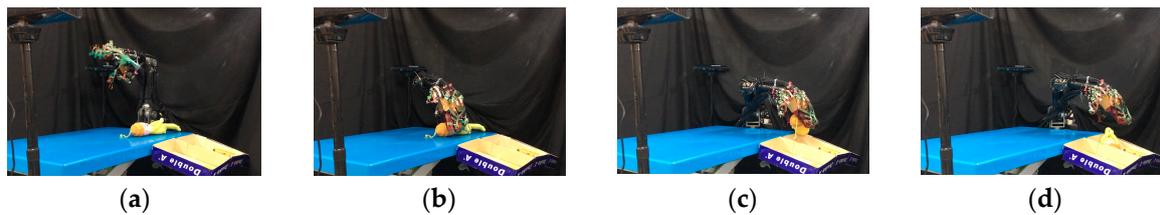


Figure 51. A procedure of grasping the fifth object: (a) The target object (low stiffness category); (b) Collision-free paths for grasping; (c) RRT-Connect algorithm; (d) Successfully separated and placed in different boxes.

As also shown in Figure 52, all the objects on the table were successfully separated and placed in different boxes based on the stiffness of the object.



Figure 52. Classification results.

6. Conclusions

The aim of those experiments was to train a robot hand-arm system to grasp unknown objects. Grasping an object is a simple task for a human, who can grasp and manipulate any object whether he has seen it or not. We want a robot that can work in a human environment in a similar manner to grasp a variety of objects of different materials, shapes, and sizes in a cluttered scene. Therefore, we focused on two issues. The first issue was the methodology for setting the real-time grasping force of the fingers when grasping an object with unknown stiffness, weight, and friction. The second issue was on-line sorting objects in a cluttered scene.

When the task is to grasp an unknown object, it is the most concern to avoid dropping it due to slipping, which is considered a serious error because of breakage of the object or unpredictable disasters. What is more, crushing objects or grasping them with excessive force should also be avoided. In order to grasp an unknown object with an appropriate force, we used multi-sensor inputs and sensor fusion technology to detect slippage and measure the stiffness of objects. The experiments demonstrated that multi-sensor information can ensure that the robot hand grasp unknown objects safely and make the robot hand's motion more like that of a human.

Physically interacting to improve the understanding of an environment is a natural behavior for humans, and based on this, we proposed an algorithm that combines perception and manipulation to enable a robot to sort objects in a cluttered space according to one specific property (in this case, stiffness). In contrast to pure visual perceptual approaches, our method can quickly rearrange objects. Our experiments show that sorting is more viable and reliable when using this method to deal with a quantity of unknown objects.

Author Contributions: Conceptualization, S.-Q.J., M.-B.H. and H.-P.H.; methodology, S.-Q.J., M.-B.H. and H.-P.H.; software, S.-Q.J. and M.-B.H.; validation, S.-Q.J. and M.-B.H.; formal analysis, S.-Q.J. and M.-B.H.; investigation, S.-Q.J. and M.-B.H.; resources, H.-P.H.; data curation, S.-Q.J. and M.-B.H.; writing—original draft preparation, S.-Q.J. and M.-B.H.; writing—review and editing, M.-B.H. and H.-P.H.; visualization, M.-B.H. and H.-P.H.; supervision, M.-B.H. and H.-P.H.; project administration, H.-P.H.; funding acquisition, H.-P.H.

Funding: This research was funded by HIWIN Technologies Corporation, Taiwan, grant number 104-S-A13. This work was partially supported by the Ministry of Science and Technology (MOST), Taiwan, grant number 107-2221-E-002-176-MY3.

Acknowledgments: The authors would like to thank the Ministry of Science and Technology (MOST).

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

References

1. Huang, H.-P.; Yan, J.-L.; Huang, T.-H.; Huang, M.-B. IoT-based networking for humanoid robots. *J. Chin. Inst. Eng.* **2017**, *40*, 603–613. [CrossRef]
2. Saudabayev, A.; Varol, H.A. Sensors for robotic hands: A survey of state of the art. *IEEE Access* **2015**, *3*, 1765–1782. [CrossRef]
3. Huang, H.-P.; Yan, J.-L.; Cheng, T.-H. Development and fuzzy control of a pipe inspection robot. *IEEE Trans. Ind. Electron.* **2010**, *57*, 1088–1095. [CrossRef]
4. Huang, M.-B.; Huang, H.-P.; Cheng, C.-C.; Cheng, C.-A. Efficient grasp synthesis and control strategy for robot hand-arm system. In Proceedings of the 11th IEEE International Conference on Automation Science and Engineering (CASE), Gothenburg, Sweden, 24–28 August 2015.
5. Lee, W.-Y.; Huang, M.-B.; Huang, H.-P. Learning robot tactile sensing of object for shape recognition using multi-fingered robot hands. In Proceedings of the 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Lisbon, Portugal, 28 August–1 September 2017.
6. Liu, Y.-R.; Huang, M.-B.; Huang, H.-P. Automated grasp planning and path planning for a robot hand-arm system. In Proceedings of the 11th IEEE/SICE International Symposium on System Integration (SII), Paris, France, 14–16 January 2019.
7. Huang, M.-B.; Huang, H.-P. Innovative human-like dual robotic hand mechatronic design and its chess-playing experiment. *IEEE Access* **2019**, *7*, 7872–7888. [CrossRef]
8. Rusu, R.B. Semantic 3D object maps for everyday manipulation in human living environments. *Künstl. Intell.* **2010**, *24*, 345–348. [CrossRef]
9. Huebner, K.; Ruthotto, S.; Kragic, D. Minimum volume bounding box decomposition for shape approximation in robot grasping. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA), Pasadena, CA, USA, 19–23 May 2008.
10. Lei, Q.; Chen, G.; Wisse, M. Fast grasping of unknown objects using principal component analysis. *AIP Adv.* **2017**, *7*, 095126. [CrossRef]
11. Liu, Z.; Kamogawa, H.; Ota, J. Fast grasping of unknown objects through automatic determination of the required number of mobile robots. *Adv. Robot.* **2013**, *27*, 445–458. [CrossRef]
12. Faria, D.R.; Dias, J. 3D hand trajectory segmentation by curvatures and hand orientation for classification through a probabilistic approach. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), St. Louis, MO, USA, 10–15 October 2009.
13. Hirano, Y.; Kitahama, K.-I.; Yoshizawa, S. Image-based object recognition and dexterous hand/arm motion planning using RRTs for grasping in cluttered scene. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Edmonton, AB, Canada, 2–6 August 2005.
14. Yang, K.; Sukkarieh, S. 3D smooth path planning for a UAV in cluttered natural environments. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, 22–26 September 2008.
15. Bicchi, A.; Kumar, V. Robotic grasping and contact: A review. In Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA, USA, 24–28 April 2000.
16. Murray, R.M.; Li, Z.; Sastry, S.S. Multifingered hand kinematics. In *A Mathematical Introduction to Robotic Manipulation*, 1st ed.; CRC Press: Boca Raton, FL, USA, 1994; pp. 211–240.
17. Roa, M.A.; Suárez, R. Grasp quality measures: Review and performance. *Auton. Robot.* **2015**, *38*, 65–88. [CrossRef] [PubMed]
18. Zheng, Y. An efficient algorithm for a grasp quality measure. *IEEE Trans. Robot.* **2013**, *29*, 579–585. [CrossRef]
19. Ferrari, C.; Canny, J. Planning optimal grasps. In Proceedings of the 1992 IEEE International Conference on Robotics and Automation (ICRA), Nice, France, 12–14 May 1992.
20. Boyd, S.P.; Wegbreit, B. Fast computation of optimal contact forces. *IEEE Trans. Robot.* **2007**, *23*, 1117–1132. [CrossRef]
21. WACOH-TECH Inc. Available online: <https://wacoh-tech.com/products/mudynpick/maf-3.html> (accessed on 3 September 1988).

22. Altman, N.S. An introduction to kernel and nearest-neighbor nonparametric regression. *Am. Stat.* **1992**, *46*, 175–185.
23. Kozma, L. k Nearest Neighbors algorithm (kNN). Helsinki University of Technology, Special Course in Computer and Information Science. Available online: <http://www.lkozma.net/knn2.pdf> (accessed on 20 February 2008).
24. Shaw-Cortez, W.; Oetomo, D.; Manzie, C.; Choong, P. Tactile-based blind grasping: A discrete-time object manipulation controller for robotic hands. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1064–1071. [[CrossRef](#)]
25. Johansson, R.S.; Westling, G. Roles of glabrous skin receptors and sensorimotor memory in automatic control of precision grip when lifting rougher or more slippery objects. *Exp. Brain Res.* **1984**, *56*, 550–564. [[CrossRef](#)] [[PubMed](#)]
26. Deng, H.; Zhong, G.; Li, X.; Nie, W. Slippage and deformation preventive control of bionic prosthetic hands. *IEEE/ASME Trans. Mech.* **2017**, *22*, 888–897. [[CrossRef](#)]
27. Teshigawara, S.; Tadakuma, K.; Ming, A.; Ishikawa, M.; Shimojo, M. High sensitivity initial slip sensor for dexterous grasp. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA), Anchorage, AK, USA, 3–7 May 2010.
28. Heil, C.; Colella, D. Dilation equations and the smoothness of compactly supported wavelets. In *Wavelets: Mathematics and Applications*, 1st ed.; CRC Press: Boca Raton, FL, USA, 1993; pp. 163–202.
29. Boggess, A.; Narcowich, F.J. Other wavelet topics. In *A First Course in Wavelets with Fourier Analysis*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2009; pp. 250–272.
30. Tremblay, M.R.; Cutkosky, M.R. Estimating friction using incipient slip sensing during a manipulation task. In Proceedings of the 1993 IEEE International Conference on Robotics and Automation (ICRA), Atlanta, GA, USA, 2–6 May 1993.
31. Kappassov, Z.; Corrales, J.-A.; Perdereau, V. Tactile sensing in dexterous robot hands—Review. *Robot. Auton. Syst.* **2015**, *74*, 195–220. [[CrossRef](#)]
32. Jähne, B. Image representation. In *Digital Image Processing*, 6th ed.; Springer: Berlin, Germany, 2005; pp. 41–62.
33. Alcazar, J.A.; Barajas, L.G. Estimating object grasp sliding via pressure array sensing. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA), Saint Paul, MN, USA, 14–18 May 2012.
34. Vapnik, V.N. Constructing learning algorithms. In *The Nature of Statistical Learning Theory*, 1st ed.; Springer: New York, NY, USA, 1995; pp. 119–156.
35. Cheng, C.-A.; Huang, H.-P. Learn the Lagrangian: A vector-valued RKHS approach to identifying Lagrangian systems. *IEEE Trans. Cybern.* **2016**, *46*, 3247–3258. [[CrossRef](#)] [[PubMed](#)]
36. Cheng, C.-A.; Huang, H.-P.; Hsu, H.-K.; Lai, W.-Z.; Cheng, C.-C. Learning the inverse dynamics of robotic manipulators in structured reproducing kernel Hilbert space. *IEEE Trans. Cybern.* **2016**, *46*, 1691–1703. [[CrossRef](#)] [[PubMed](#)]
37. Lo, S.-Y.; Cheng, C.-A.; Huang, H.-P. Virtual impedance control for safe human-robot interaction. *J. Intell. Robot. Syst.* **2016**, *82*, 3–19. [[CrossRef](#)]
38. LIBSVM: A Library for Support Vector Machines. 2001. Available online: <https://www.csie.ntu.edu.tw/~cjlin/libsvm> (accessed on 1 September 2001).

