ROBOMECH Journal

RESEARCH ARTICLE

Open Access

Tactile sensor-less fingertip contact detection and force estimation for stable grasping with an under-actuated hand

Ha Thang Long Doan[1], Hikaru Arita[2] and Kenji Tahara[2*]

Abstract

Detecting contact when fingers are approaching an object and estimating the magnitude of the force the fingers are exerting on the object after contact are important tasks for a multi-fingered robotic hand to stably grasp objects. However, for a linkage-based under-actuated robotic hand with a self-locking mechanism to realize stable grasping without using external sensors, such tasks are difficult to perform when only analyzing the robot model or only applying data-driven methods. Therefore, in this paper, a hybrid of previous approaches is used to find a solution for realizing stable grasping with an under-actuated hand. First, data from the internal sensors of a robotic hand are collected during its operation. Subsequently, using the robot model to analyze the collected data, the differences between the model and real data are explained. From the analysis, novel data-driven-based algorithms, which can overcome noted challenges to detect contact between a fingertip and the object and estimate the fingertip forces in real-time, are introduced. The proposed methods are finally used in a stable grasp controller to control a triple-fingered under-actuated robotic hand to perform stable grasping. The results of the experiments are analyzed to show that the proposed algorithms work well for this task and can be further developed to be used for other future dexterous manipulation tasks.

Keywords Fingertip contact detection, Fingertip force estimation, Multi-fingered hand, Externally sensorless grasping, Grasping, Manipulation

Background

In the robotic manipulation field, grasping using under-actuated hands has recently been receiving more attention [1–4]. This is because, by relieving the over-constraint problem, mechanical compliance can be achieved, and the power grasping controller becomes simple. In addition, a lower number of actuators means that the robotic hand is more energy- and cost-efficient and can be lighter compared to a fully actuated hand. This paper focuses on one common type of under-actuated hand, a linkage-based mechanism, which facilitates serial and parallel links to transmit high actuator power to the fingertip to execute a large force to grasp large/heavy objects [5].

During the finger bending process, a common linkage-based under-actuated mechanism operates using one or several mechanical parts that act as "rotational limiters" (e.g., breaks or stoppers [6]), which lock some of the links in the finger base from moving in a certain direction, while forcing other links at the upper part to move, causing the tip of the finger to bend [7, 8] (see Fig. 1). By utilizing a self-locking mechanism, the finger can grasp objects in many situations, even under significant object pose and size uncertainties. Hence, adaptive or power grasping with the fingers automatically

*Correspondence:
Kenji Tahara
tahara@ieee.org
[1] Department of Mechanical Engineering, Graduate School of Engineering, Kyushu University, 744 Motooka Nishi-ku, Fukuoka, Japan
[2] Department of Mechanical Engineering, Faculty of Engineering, Kyushu University, 744 Motooka Nishi-ku, Fukuoka, Japan

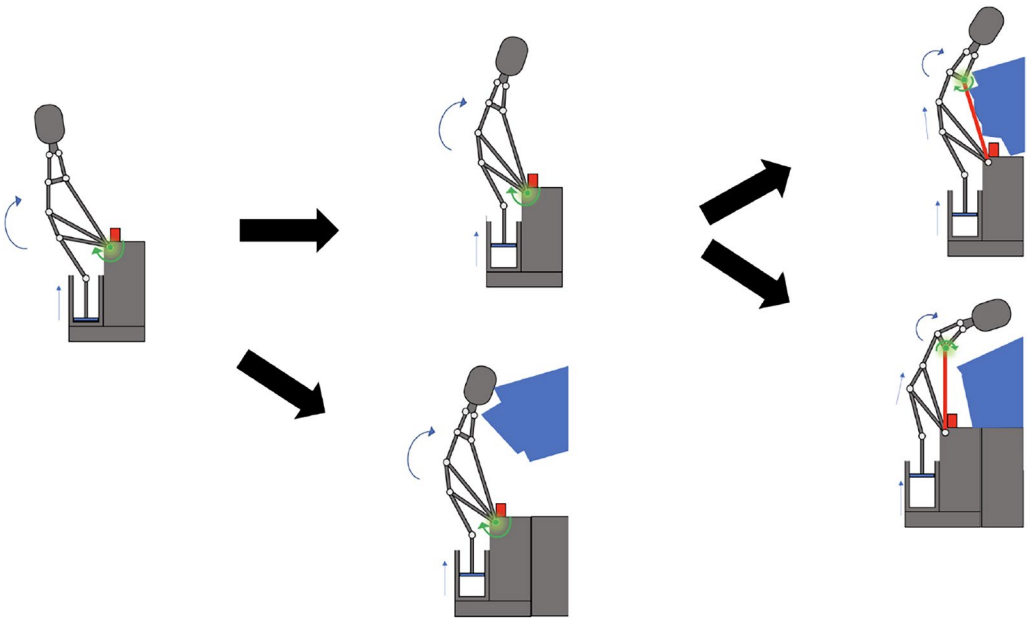


Fig. 1 Linkage-based under-actuated self-locking hand mechanism

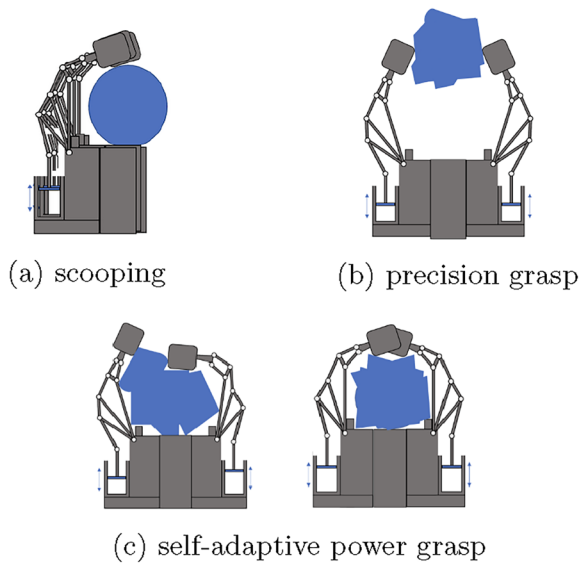


Fig. 2 Tasks using an under-actuated hand

wrapping around an object without specifying a desired fingertip position during control can be realized (see Fig. 2c). Besides the indicated advantages of a linkage-based under-actuated hand, this particular mechanism also allows the hand to be easily bent to perform other tasks, such as scooping (see Fig. 2a) or simple object manipulation without complex sensing [9].

However, the use of self-locking mechanisms in linkage-based under-actuated hands create passive shifts that lead to nonlinearity in the kinematics/dynamics of the fingers during the finger-bending process, making it difficult to model, as well as to obtain necessary information for precision grasping, such as the detection of contact between the fingertip and object to switch the control phase from approaching to grasping, or the estimation of fingertip force to perform precision grasping. Therefore, although there are several existing studies on fingers with a similar mechanism, the majority of their focus is simple self-adaptive power grasping (see Fig. 2c) [10, 11]. Due to such challenges, while dexterous manipulation methodologies for fully-actuated hands have already been introduced throughout the years, it is still challenging for linkage-based under-actuated hands to realize even a simple task such as stable object grasping using the fingertips. To allow linkage-based under-actuated hands to realize more complicated precision grasping tasks, the focus of this paper is developing new methodologies to obtain the required information to perform stable grasping, which is the foundation task for dexterous manipulation.

In recent years, to address the problem of uncertainties in grasping/manipulation, vision and sensor-based methodologies to obtain as much information as possible have become very common, and experiments are mainly performed in an ideal laboratory environment [12–15]. Although the indicated controllers can be used in very specific situations (e.g., in a location with enough lighting and no significant or frequent changes in the environment), they are still not flexible enough to be used for a

variety of other tasks in the industry (e.g., outdoor tasks, rough use). The reasons behind this vary, with the most common reasons being that the aforementioned sensors and controllers are still under development and the environment can affect the sensor data quality, which can affect the robustness of the method. Camera sensors, which are used very commonly to collect information about objects to be grasp, do not work well in extreme lighting conditions or on reflective objects. On the other hand, special types of tactile sensors used to obtain the fingertip contact point and fingertip force still have problems with robustness, noise, accuracy, and drift, and are difficult to maintain [16], so they are not ready to be used in some uncertain situations (e.g., grasping unknown objects, in unstructured environments, grasping objects that can damage the tactile sensors). Rather, the majority of manipulators/robots in the industry still have robust and widely used sensors attached inside the hand, far from the tip, and necessary information is obtained by transforming the sensor signals. For that reason, to satisfy industry demands, the focus of this paper is to develop a methodology to transform the internal sensor data in real-time so that unknown information can be obtained with enough accuracy for manipulation tasks. Furthermore, this also means that it is better not to rely entirely on those devices when manipulating unknown objects or in an unstructured environment, and external sensor-less controllers still need further investigation.

For the aforementioned reasons, the significant challenge in in-hand manipulation using a linkage-based under-actuated hand is to keep a grasped object stable while performing dexterous manipulation tasks in an unstructured environment, in which prior information about the object is missing, sensor data are prone to be noisy, and preparing additional special sensing devices to support this operation is not a solution. Therefore, this paper focuses on developing a methodology to acquire the required information to utilize the previously developed stable grasping controller, namely the contact detection between the fingertip and object, as well as the fingertip force, while overcoming the challenges of (1) noisy internal sensor data, (2) non-linear kinematics/dynamics of a linkage-based mechanism, and (3) not using external sensors to support the task.

Related work

External sensorless grasping/manipulation

Regarding research on external sensorless methods for dexterous manipulation, in recent decades, the common method is to apply an analytics model-based approach. This technique uses a robot model with many assumptions to compute a mathematic relation between known and necessary information to control the robot. This also allows researchers to understand the phenomenon behind the dynamics/kinematics of the robot and to simulate what is likely to happen without risking damaging the real robot. Using this approach, Arimoto et al. introduced a new modeling and analytical-mechanics approach to simulate the grasping process without using any information regarding fingertip contact point and object mass [17–19]. Then, in subsequent work, significant research effort has been applied to the idea to further develop various control methodologies. Regarding stable grasping with a triple-fingered fully actuated hand, Tahara et al. [20] developed an external sensorless post-contact controller to stably grasp objects. However, state-of-the-art methods still face many challenges when modeling multi-fingered hands. For linkage-based under-actuated fingers with nonlinear complex kinematics/dynamics, which are difficult to model, these techniques face additional problems with inaccuracy, making them unsuitable for precision grasping tasks. Additionally, to automate the whole grasping process with these post-contact controllers, switching signals or contact between the fingertips and object should be detected in real-time, which is challenging for an under-actuated hand to realize without using additional sensors. Facing somewhat similar problems, Ozawa et al. [21] also discussed a "force-sensorless fault tolerant detection and switching control" for a tendon-based under-actuated hand. For linkage-based under-actuated hands, external sensorless methods are still not available for fingertip contact detection and force estimation. For these reasons, while many controllers for various types of robots have been studied, the majority cannot be applied directly to a linkage-based under-actuated hand, and methods to obtain the necessary information to regulate the robot using feedback need to be further investigated.

Precision grasping/dexterous manipulation with an under-actuated hand

When addressing uncertainties in controlling under-actuated hands, many studies have focused on using only an end-to-end learning framework with little or no use of the robot's physics knowledge [13, 15]. The majority of these experiments are carried out in an ideal environment with various types of external sensors to gather as much information as possible for the robot to make decisions. Hence, an end-to-end machine learning framework is not the ideal solution for stable grasping in the case of no additional external sensors. However, in the last decade, some researchers have addressed this challenge from a different point of view. To make under-actuated fingers useful for dexterous manipulation tasks, Liarokapis and Dollar [22–24] presented methodologies that combine learning with analytics models for dexterous

manipulation to overcome the difficulties of controlling a tendon-driven under-actuated hand. The use of an analytics model in the learning framework significantly reduced the amount of sensor data required for learning, and its effectiveness was also demonstrated. Yet, similar approaches have not been applied to a linkage-based under-actuated hand. On the other hand, significant improvement in model-based controllers combined with reinforcement learning for manipulation using the entire robotic arm has also been demonstrated [25, 26]. Thus, while only a few papers have focused on this approach, combining an analytics model and learning has been demonstrated to be an effective method to address the challenges this paper focuses on.

Data-driven based stable grasping method

In [27], the idea of using a well-known clustering algorithm in the data science field, the density-based spatial clustering of applications with noise (DBSCAN) algorithm [28], to detect "pre-contact" and "post-contact" data points was discussed. The idea of choosing suitable parameters for the algorithm, as well as how to simplify the algorithm for real-time use is also briefly discussed in this paper. Then, in [29], the preliminary results of stable grasping with a data-driven-based controller for each finger were introduced, which can benefit from the use of contact detection and fingertip force estimation methods to regulate the grasping process under disturbance. In this paper, using the proposed ideas as a foundation, two novel algorithms to detect contact and estimate the force the fingertips exert on an object in real-time and a detailed explanation of a stable grasping controller for the whole under-actuated hand are presented.

Method

In this paper, to have an external sensor-less stable grasping controller for an under-actuated hand, a hybrid model-based and data-driven approach is used, so that the advantages of both can be utilized while reducing the disadvantages of each. The structure of this methodology is shown in Fig. 3. The study starts with collecting a large amount of data from the internal sensors of a robotic hand during finger bending operations. Next, using a robot statics model and field knowledge, the meaning of the collected data is explained. Using the analysis, the difference between the analytics model with various assumptions and the phenomenon observed in the real robot is determined. Then, novel data-driven-based algorithms, which are suitable to overcome the problem with modeling such phenomenon, are proposed. Such methods are used to replace the specific parts in the model which do not represent real behavior well. Finally, using the proposed data-driven methods in a model-based controller, stable grasping experiments are conducted, the experimental data are collected for analysis, and the cycle is repeated.

Paper structure

The subsequent parts of this paper are structured as follows: The "Problem Statement" section explains the design and the background of the robotic hand used in the experiment, the problem this paper is focusing on, and the generalization of the proposed method to similar hands. The "Data Collection and Analysis" section presents the collected time series data from internal sensors during the finger-bending process, and explains the phenomenon observed. The "Fingertip Contact Detection" section introduces the method used to learn the behavior of the collected data to find suitable parameters for the DBSCAN algorithm to detect contact data points,

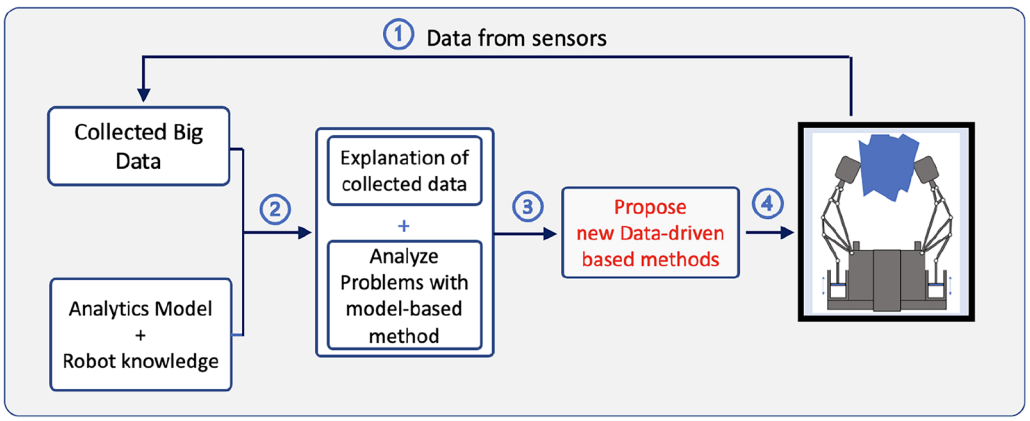


Fig. 3 Proposed approach

and the novel algorithm which uses learned information to detect contact in real-time. The "Fingertip Force Estimation" section proposes a new algorithm, which utilizes the Fingertip Contact Detection algorithm output to estimate the fingertip force during grasping. The "External Sensor-less Stable Grasping Technique using a Virtual Frame" section explains the stable grasping technique applied in this paper. In the "Stable Grasp Controller for the Jinki Hand" section, a controller for each finger, as well as the controller for the whole robotic hand are presented. Using this controller, stable grasping experiments are carried out and discussed in the "Stable Grasping Experiment" section, and the data collected are analyzed to evaluate the proposed algorithms. The paper ends with the "Conclusion" section summarizing the content of the paper and a discussion of the methodology results and how it will be used in future work.

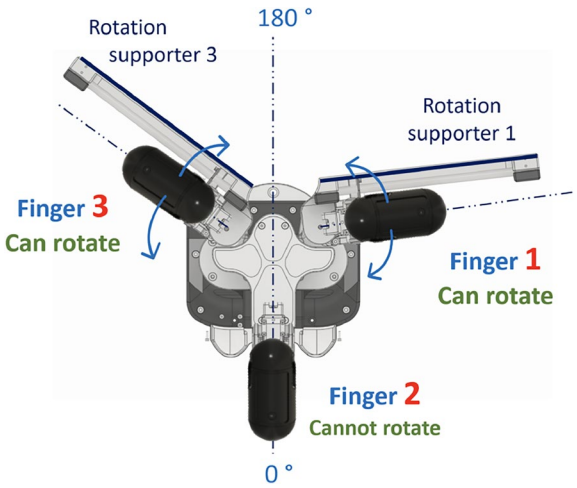


Fig. 5 Jinki Hand top view

Problem statement

In this experiment, a triple-fingered under-actuated robotic hand (the Jinki Hand) (see Fig. 4) made by our collaborator, Man–Machine Synergy Effectors Inc. (MMSE) [30], is used. MMSE has been developing humanoid robots to perform dangerous and dexterous manipulation tasks in unstructured industry environments, and the Jinki Hand was developed to be attached to those humanoid robots. The Jinki Hand has three linkage-based under-actuated fingers with the same design: an index (Finger 1), thumb finger (Finger 2), and middle finger (Finger 3) (see Fig. 5). The hand was designed so that Fingers 1 and 3 can freely rotate from 0° to 100° with support from two rotation supporters, while Finger 2 remains fixed. The configuration of each finger is shown in Fig. 6. Each finger has a lead screw located at the bottom which connects with a motor. By applying a voltage to the motor, the lead screw moves up and down in the vertical direction. From the motor encoder, the location of the lead screw can be determined. Therefore, the use of the motor and lead screw in this design works as a linear actuator to bend each finger. A rotational limiter is designed to be at a location to lock some specific links (see Fig. 1). In each finger, a load cell is attached to the link that connects the actuator with the upper part of the finger to measure the internal force in this link in real-time.

It is important to note that the location of the load cell sensor in the Jinki Hand is far from the fingertip, and with this complex configuration, creating a precise analytical model to obtain the relationship between the load cell value and fingertip force is challenging. This paper focuses on the linkage-based under-actuated hands with a similar self-locking mechanism, in which internal force and position sensors can be attached to the link inside the robot for the data-driven-based methods to be applied. More specifically, each time the fingertip touches the object, there should be a change in the internal force that can be captured by a force sensor. Our methodologies then utilize such data and overcome other challenges such as sensor noise, and discrete shifts in the kinematics/dynamics of the hand to detect contact. A force sensor should also be placed in the location where the statics relation between its data and the fingertip force can be derived. Then, the novel algorithm can obtain unknown

Fig. 4 Jinki Hand

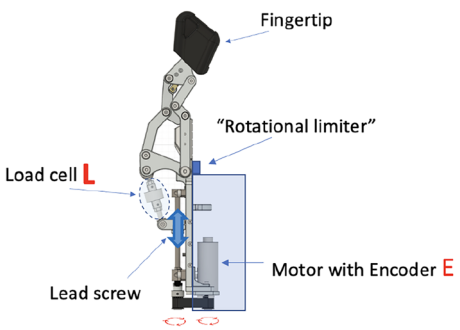


Fig. 6 Jinki Hand finger configuration

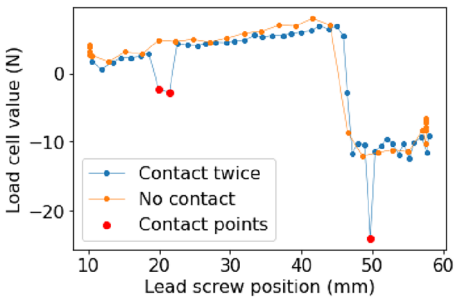


Fig. 7 Load cell - lead screw position data during one finger-bending process

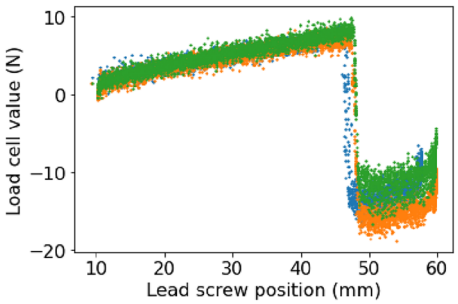


Fig. 9 Difference in the data collected from three fingers with the same configuration

terms necessary to derive the fingertip force. One position sensor should be at the location where the kinematics relation between its data and the fingertip position during precision grasping can also be formulated. As long as those conditions are met, similar hands with the same mechanism but different specifications, including different numbers and sizes of links or with more rotational locks, can also apply the generalized novel algorithms.

In this paper, we focus on developing a stable grasping control system for the linkage-based under-actuated hands. Our approach is to utilize the external sensorless fingertip grasping technique that has already been validated to be able to keep objects stable without utilizing object properties in the controller [20]. To keep the object stable, the technique requires all of the fingertips to be in contact with the unknown object, and then control the fingertip force in a feedforward manner without knowing the exact contact location and force of each fingertip. In other words, to enable such a technique to a linkage-based under-actuated hand, only obtaining the boolean information of whether each fingertip contacted the object or not, and a method to roughly control the fingertip force, are necessary. However, such an approach is difficult to realize on a linkage-based under-actuated hand with no tactile sensors. Therefore, our solution is to adjust the controller and introduce novel fingertip contact detection and force estimation algorithms for feedback control to execute the stable grasping process. Thus, it is important to note that the term "fingertip contact detection" in this paper refers to the process of obtaining the boolean information of fingertip contact/non-contact. Our work is to confirm that such an approach allows the linkage-based under-actuated hands to execute the previously proposed stable grasping technique.

Data collection and analysis

Data collection

First, the data from the internal sensors of the robotic hand, including the position encoder and load cell at each finger, were collected during one finger-bending process, which is the process from when the finger is straightened until it is fully bent without contacting the object. The finger-bending process was performed again, but this time we touched the fingertip twice during the process (once before self-locking occurred, and once after) (see Fig. 7). The experiment is conducted to observe the changes in the internal sensors data when the fingertip is touched. Subsequently, for each finger, the finger-bending process was carried out 200 times (see Fig. 8), which is assumed to be enough to record most of the possible internal sensor behavior during the finger-bending process. The data are cleaned so that duplicated values are removed and the data are

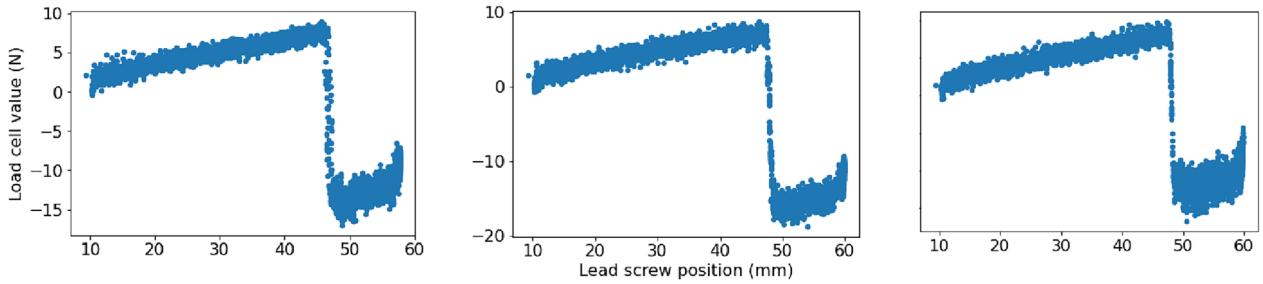


Fig. 8 Load cell - lead screw position data during 200 finger-bending processes

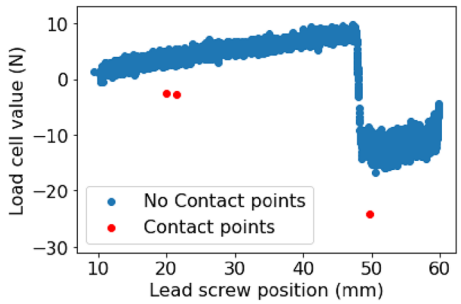


Fig. 10 Deviation of contact data points from the non-contact point "area"

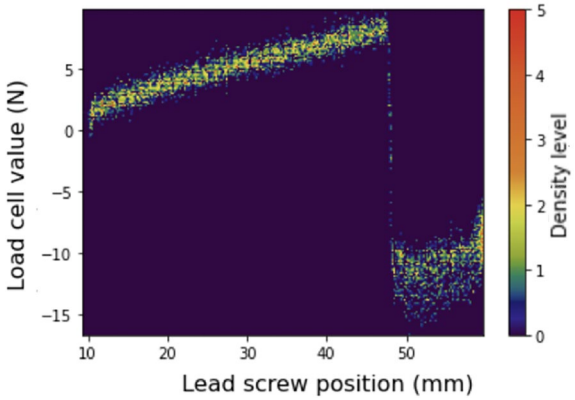


Fig. 11 Data density of each 200 finger-bending dataset

rounded to a significant figure suitable for the sensor resolution. It is important to note that the load cell and lead screw position encoder values are both initialized at the start of the experiment, when the finger is fully straightened. Hence, the data obtained are with respect to the initialized value.

Analysis of collected data

Nonlinear behavior of under-actuated fingers

In each of the three subfigures in Fig. 8, a shift in the load cell data when changing the lead screw position around 45 mm can be observed. The same phenomenon can also be seen in Fig. 7, in both the "No contact" and "Contact twice" experiments.

The shift in the load cell sensor data that occurs during each finger-bending process, as explained above, is due to the use of one rotational limiter in the bending mechanism of the linkage-based under-actuated finger, and the shift represents the change in the internal force when the brake starts exerting force on the link for the self-locking mechanism.

Lack of replication in the same finger bending process and same finger configuration

From Fig. 8, it can be seen that the process lacks replication in internal sensor data even though the same finger movement was carried out 200 times for each finger. This can be explained as the effect of sensor noise, drift, calibration error, hand vibration, and many other external factors. In industrial use, these factors cannot be fully controlled, and the sensor data are expected to vary even more when attached to the robot's arm. In addition, since the exact lead screw position where the self-locking occurred, even for the same finger, differs in every finger bending process, the nonlinear behavior becomes even more challenging to solve. Moreover, while having the same design, due to slight errors in the robotic hand manufacturing process, the data collected are different for the three fingers (see Fig. 9). Thus, using the robot design to create an analytics model could result in a highly inaccurate controller.

Deviation of the contact data from the "non-contact zone"

In Fig. 7, it is obvious that the contact data points deviate completely from where the data usually appear on the graph. The visualization of this idea can be seen in Fig. 10. In addition, when plotting the data density graph of each 200 finger-bending data collected (the density is the data point per pixel in the figure, see Fig. 11), the result showed that in these collected data, the non-contact zone has much higher density. Thus, a data-driven-based classification method using data density characteristics can be used to detect contact and non-contact data points.

Fingertip contact detection

Contact and non-contact data points detection using the DBSCAN algorithm

In [27], the concepts and use of the DBSCAN algorithm in the robotic manipulation field are introduced. The DBSCAN algorithm goes through every data point in the input dataset and detects its cluster using the following rule: a data point that has more than *MinPts* data points in the *Eps* neighborhood (within the distance *Eps*) belongs to the same cluster as those points (see Fig. 12). The metric for the distance between 2 data points used here is the Euclidean distance in the data science field. After analyzing all data points, the data points which do not belong to any cluster are detected as outliers [28]. In the "Data Collection and Analysis" section, 200 finger bending data for each finger were collected, which are assumed to represent all of the internal sensor data behavior when the fingertip is not in contact with the object (non-contact representative datasets). As a result,

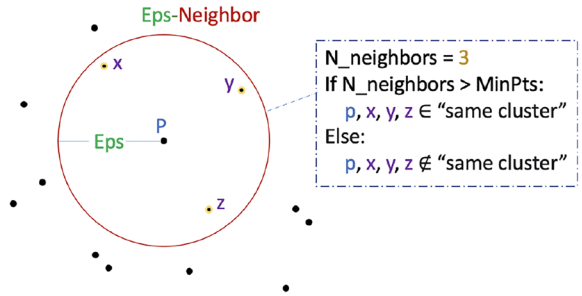


Fig. 12 Each loop in the DBSCAN algorithm

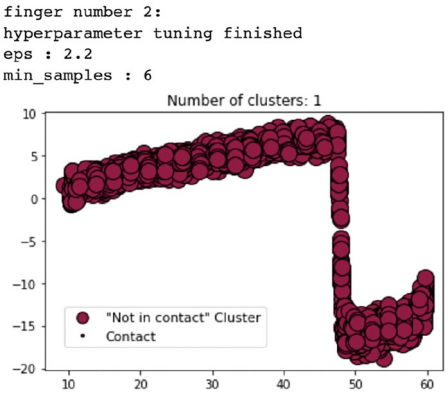


Fig. 13 Tuning result example

it is necessary that each DBSCAN algorithm tuning detects all points in the respective non-contact representative dataset to be in the same "non-contact cluster". In addition, it is also necessary for the DBSCAN algorithm to be sensitive enough to detect the contact points as outliers. For the parameter tuning process, each of the non-contact representative datasets is used to determine the suitable set of DBSCAN parameters for each finger.

The DBSCAN parameter tuning algorithm for contact data detection (see Algorithm 1) requires three inputs: D is the representative dataset of the finger currently tuning, 1 to *MaxPts* is the tuning range of *MinPts*, and *FoS* is the safety factor. This algorithm outputs the tuned *tEps* and *tMinPts* values, which can be used for the DBSCAN algorithm to cluster non-contact and contact data points. The goal of the tuning process is to obtain the set of parameters that can make the DBSCAN algorithm accurate and sensitive to detect contact data points as outliers, and it is designed as follows:

For the tuning process on D, initially (lines 1 to 3), *MinPts* is set to *MaxPts* and *Eps* is first set to *DeltaEps*, which is the higher resolution between the lead screw position encoder and the load cell sensor. *DeltaEps* should not be smaller than this value because the sensors do not have enough resolution for sensing it. To start the tuning loop, loop condition *Continue* is initialized to TRUE (line 4). In each tuning loop (lines 5 to 19), first, the previously assigned parameters *MinPts* and *Eps* are checked using the DBSCAN algorithm on D (line 6).

If the current set of parameters cannot detect all non-contact representative dataset D as "non-contact" (the number of clusters $N_{cluster} \neq 1$ and the number of outliers $N_{outlier} \neq 0$, line 7), these parameters need to be further adjusted. *MinPts* is adjusted first by reducing its

Algorithm 1 DBSCAN parameter tuning for contact detection

Input: $D, MaxPts, FoS$
1: $MinPts \leftarrow MaxPts$
2: $DeltaEps \leftarrow Max(LeadScrewResolution, LoadCellResolution)$
3: $Eps \leftarrow DeltaEps$
4: $Continue \leftarrow$ TRUE
5: **while** $Continue =$ TRUE **do**
6: $N_{cluster}, N_{outlier} \leftarrow DBSCAN(D, Eps, MinPts)$
7: **if** $N_{cluster} \neq 1$ or $N_{outlier} \neq 0$ **then**
8: **if** $MinPts > 1$ **then**
9: $MinPts \leftarrow (MinPts - 1)$
10: **else if** $MinPts = 1$ **then**
11: $Eps \leftarrow (Eps + DeltaEps)$
12: $MinPts \leftarrow MaxPts$
13: **end if**
14: **else if** $N_{cluster} = 1$ & $N_{outlier} = 0$ **then**
15: $Continue \leftarrow$ FALSE
16: $tEps \leftarrow RoundUp(Eps * FoS)$
17: $tMinPts \leftarrow RoundDown(MinPts * FoS)$
18: **end if**
19: **end while**
Output: $tEps, tMinPts$

value by 1 if it is not already the lowest allowed value of 1 (lines 8 to 9). Otherwise, *Eps* will be increased by *DeltaEps*, and *MinPts* is reset back to its highest allowed value of *MaxPts*. Then, the adjusted values are checked in the next loop.

On the other hand, if all data points are detected as non-contact (the number of clusters $N_{cluster} = 1$ and the number of outliers $N_{outlier} = 0$), that will be the final tuning process loop (lines 14 to 15). In an ideal case, the representative dataset can represent all of the possible data points during pre-contact. However, in reality, this is not the case. Therefore, a safety factor *FoS* is applied to the parameters in the final tuning loop *MinPts* and *Eps* results, then the values are rounded to obtain the final output, *tEps* and *tMinPts* (lines 16 to 17). In other words, the user of this algorithm can slightly self-adjust the tuned parameters to further improve the accuracy and sensitiveness of the algorithm, depending on the situation. Thus, using this algorithm, the parameters necessary for the DBSCAN algorithm to detect contact data points can be acquired. An example of the tuning result is shown in Fig. 13. The same process is performed for all three fingers to find suitable parameters for each finger.

The tuning process enables the algorithm "learns" key information about the non-contact representative data for contact detection: There are more than *tMinPts* other data points within the *tEps* neighborhood of a non-contact data location. When comparing the real-time data with the representative dataset, if a real-time data point does not meet this requirement, it is a contact data point.

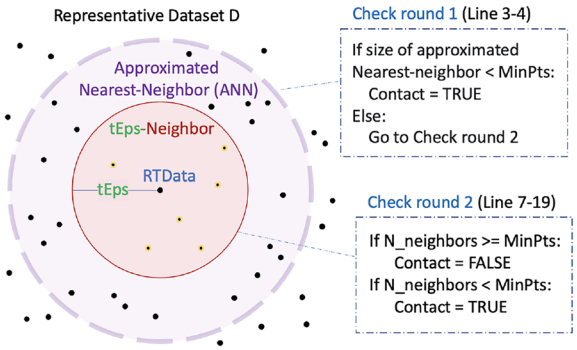


Fig. 14 Real-time Fingertip Contact Detection algorithm

Real-time fingertip contact detection algorithm

In the "Contact and non-contact data points detection using the DBSCAN algorithm" section, a method using the DBSCAN algorithm to detect contact and non-contact data points is presented. However, since the original DBSCAN algorithm was designed to go through every data point in the input dataset each time the algorithm is called, for fingertip contact detection, this is a waste of computing resources. The main goal of controlling an under-actuated hand is to stably grasp objects, hence, the contact detection algorithm should be simplified to be used in real-time with low memory usage. The fingertip contact detection, inspired by the DBSCAN algorithm, is shown in Algorithm 2, and a visualization of the algorithm is shown in Fig. 14.

Algorithm 2 Real-time fingertip contact detection

Input: $D, tEps, tMinPts, LoadcellasL, EncoderasE$
$RTData \leftarrow [E, L]$
$ANN \leftarrow Neighbor(D, RTData, tEps)$ ▷ $Neighbor()$ is the Approximate Nearest Neighbor (ANN) Search algorithm
if $Size(ANN) < tMinPts$ **then** ▷ $Size()$ gives the number of data points within the input dataset
  $Contact \leftarrow$ TRUE ▷ Check round 1
else
  $N\_neighbors \leftarrow 0$ ▷ Initialize $N\_neighbors$ to start counting true number of neighbors
  for each $Datapoint \in ANN$ **do**
    if $d(Datapoint, RTData) \leq tEps$ **then** ▷ $d()$ is the Euclidean distance function
      $N\_neighbors \leftarrow (N\_neighbors + 1)$
      if $N\_neighbors = tMinPts$ **then**
        $Contact \leftarrow$ FALSE ▷ Check round 2
        break
      end if
    end if
  end for
  if $N\_neighbors < tMinPts$ **then**
    $Contact \leftarrow$ TRUE
  end if
end if
Output: $Contact$

The real-time Fingertip Contact Detection algorithm requires five inputs: the representative dataset D, the tuned parameters *tEps* and *tMinPts*, and the real-time sensor data *RTData*, which includes load cell and lead screw position encoder data L and E. The representative dataset of each finger D is obtained as described in the "Data Collection and Analysis" section, *tEps* and *tMinPts* are obtained as described in the "Contact and non-contact data points detection using the DBSCAN algorithm" section, and the other two inputs are the signals from the internal sensor output. The algorithm works by appending the *RTData* to D, then checking the data density of *RTData* to see if it belongs to the "non-contact" cluster, or if it is an outlier, i.e., a contact point. The detail of the algorithm structure is explained as follows:

First, the real-time sensor data *RTData* is assigned as an array of E and L (line 1). Then, the subset data of D, *ANN* (Approximated Nearest-Neighbor), is obtained using a common data structure and algorithm field algorithm, Approximate Nearest Neighbor Search, to find the set of data within D which is likely to be in the *tEps-Neighbor* of the *RTData* (line 2, see Fig. 14). By doing so, when checking for the number of neighbors of *RTData* in subsequent steps, the number of data points to check is significantly decreased, hence, reducing the computing resources. It is important to note that the Approximate-NNS algorithm is used instead of the Precise-NNS algorithm because the Approximate-NNS algorithm is much faster and memory saving, making it suitable for real-time use. However, it is also necessary to choose an algorithm that over-selects the data points in the neighborhood, otherwise the algorithm will result in inaccuracy (see Fig. 14).

Here, "Check round 1" (see Fig. 14) is conducted: If the number of data points within this over-selected neighborhood *Size*(*ANN*) is less than *MinPts*, true neighbors should also be less than *MinPts*. For that reason, in this case, contact is detected (lines 3 to 4).

Otherwise, "Check round 2" (see Fig. 14) begins. The number of true neighbors *N_neighbors* is initialized to 0 (line 6), and the algorithm starts counting with a loop. In each loop, 1 data point is selected using the *ANN* and is analyzed (lines 7 to 15). If the Euclidean distance between the point and *RTData* is less than or equal to *tEps*, the point is in the true *tEps* neighborhood, hence the algorithm adds 1 more neighbor *N_neighbors* (lines 8 to 9). Right after this analysis, the counted number is checked to see if its equal to *tMinPts*. If it is, the fingertip is not in contact and there is no need to continue checking. In that case, the loop iteration is finished and is broken (lines 10 to 12). If not, the counting process continues and ends when there is nothing left to count inside the *ANN*. After checking all data points in the *ANN*, if the counted value is still not equal to *MinPts*, then the counted value is less than *MinPts* and the fingertip is detected to be in contact with the object (lines 16–18). All cases were checked, so the algorithm ends here (line 19).

Therefore, by having two check rounds designed as explained above, the algorithm can detect contact between the fingertip and object in real-time.

Fingertip force estimation

Explanation of collected data using an analytics model

In [29], a simple analytics model to explain the behavior of the internal sensor data pre and post-contact was introduced. Simply put, the key ideas that can be obtained from the model are as follows:

The load cell measures the internal force of the link inside the Jinki Hand:

1) When self-locking has not occurred, the load cell value mainly contains the internal force caused by the effect of gravity on the links and the internal force caused by the fingertip force.

$$L = L_{\text{gravity}} + L_{\text{contact}} \tag{1}$$

where L_{gravity} is the load cell value that is affected by gravity and L_{contact} is the load cell value caused by the fingertip contact force.

2) After self-locking occurs, the load cell value includes the additional internal force caused by the rotational limiter stopping the bending motion of one finger link.

$$L = L_{\text{gravity}} + L_{\text{contact}} + L_{\text{locking}} \tag{2}$$

where L_{locking} is the internal force shift caused by self-locking.

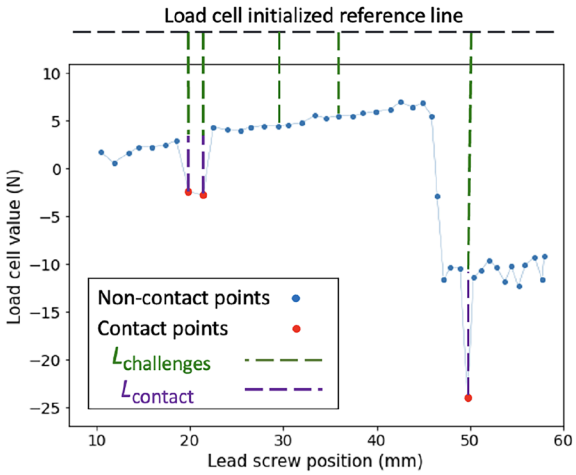


Fig. 15 Visualization of the summarized load cell value equation

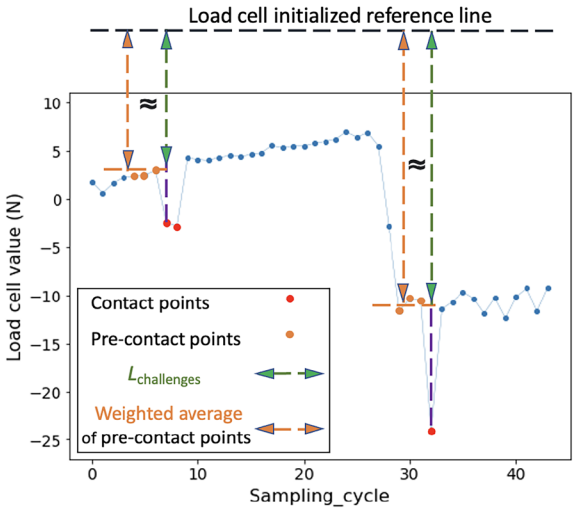


Fig. 16 Visualization of the weighted average method

Algorithm 3 Real-time fingertip force estimation

Input: $Contact, L, E$
if $T = 1$ **then** ▷ If this is the first control loop iteration
  $L_{\text{T-1}} = 0$
  $L_{\text{T-2}} = 0$
  ...
  $L_{\text{T-N}} = 0$ ▷ Initialize pre-contact load cell values
end if
if $Contact =$ FALSE **then**
  $F_{\text{contact}} = 0$
  $L_{\text{T-1}} = L$
  $L_{\text{T-2}} = L_{\text{T-1}}$ ▷ Update pre-contact load cell values
  ...
  $L_{\text{T-N}} = L_{\text{T-N+1}}$
  $ToNextIteration(L_{\text{T-1}}, L_{\text{T-2}}, ...L_{\text{T-N}})$ ▷ Send to the next control loop
else if $Contact =$ TRUE **then**
  $L_{\text{challenges}} = Weightaverage(L_{\text{T-1}}, L_{\text{T-2}}, \ ... \ L_{\text{T-N}})$
  $L_{\text{contact}} = L - L_{\text{challenges}}$
  $F_{\text{contact}} \leftarrow Statics\_model(L_{\text{contact}})$ ▷ Obtain the fingertip force using L_{contact} ▷ and the relation computed using the statics model
end if
Output: F_{contact}

Furthermore, in the "Data Collection and Analysis" section, the load cell value also includes noise and error from the manufacturing process and initialization. Thus, the above equations can be further derived:

$$L = L_{\text{challenges}} + L_{\text{contact}} \tag{3}$$

in which,

$$L_{\text{challenges}} = noise + error + \begin{cases} L_{\text{gravity}} & \text{before self-locking} \\ L_{\text{gravity}} + L_{\text{locking}} & \text{after self-locking} \end{cases} \tag{4}$$

A visualization of this equation is shown in Fig. 15. In this equation, L_{gravity} and its sum with L_{locking}, together with *noise* and *error*, is defined as $L_{\text{challenges}}$ because these values vary significantly during each finger bending process (see Fig. 8), which has already been explained in the "Data Collection and Analysis" section. From the figure, it is clear that a traditional analytics model has difficulty representing this behavior. Therefore, by using the fingertip force analysis method introduced in [29] as a foundation, Fingertip Force Estimation algorithm is proposed to overcome the challenges of obtaining $L_{\text{challenges}}$ by estimating it in real-time. Then, most of the challenges to the model parts $L_{\text{challenges}}$ can be removed (subtracted) from the load cell value L, and L_{contact} can be easily acquired:

$$L_{\text{contact}} = L - L_{\text{challenges}} \tag{5}$$

The analytics model can be used with the obtained information to estimate the fingertip force.

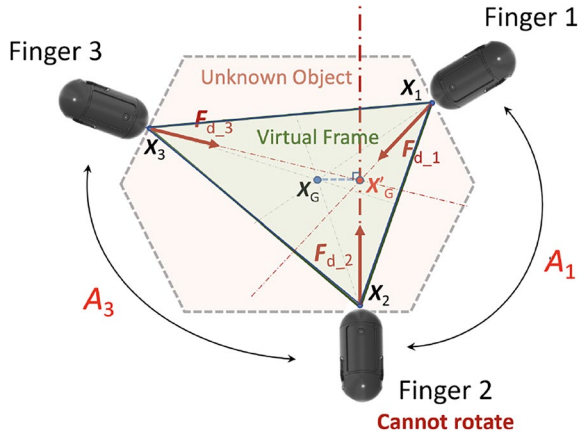


Fig. 17 Object Force/torque equilibrium condition for the triple-fingered hand

Real-time fingertip force estimation

In [29], the idea of using a basic data science field algorithm, the weighted average, to estimate L_{gravity} was proposed. Here, this method is further developed into a complete algorithm that is connected to the Fingertip Contact Detection algorithm to make it suitable for stable grasping task. The idea of using the weighted average method to estimate $L_{\text{challenges}}$ when contact is realized is as follows (see Fig. 16):

Since a precise $L_{\text{challenges}}$ is difficult to obtain offline, it should be measured/estimated in real-time. From the analysis of the collected data, it was clear that when contact is realized, $L_{\text{challenges}}$ is approximately equal to the weighted average of the load cell value of a few sampling iterations, N, right before contact is detected.

$$L_{\text{challenges}} \approx \frac{\sum_{i=1}^{N} w_i L_{\text{T-i}}}{\sum_{i=1}^{N} w_i} \tag{6}$$

Here, the weighted average method is applied to the sensor data so the noise can be partially removed and the initialization error can be significantly reduced. For this proposed idea to be used in real-time for stable grasping, it was developed into Algorithm 3. It is also important to note that the method is used to obtain fingertip contact force at the finger posture when contact is realized. During stable grasping, it is assumed that the finger posture does not change, hence, the method can be used for this task.

The algorithm requires three inputs: the *Contact* outputted from the Fingertip Contact Detection algorithm and the signals from the internal sensors load cell L and encoder E. The current control loop T should also be known to initialize N load cell values before contact, $L_{\text{T-1}}$ to $L_{\text{T-N}}$, to 0 during the beginning of the first iteration (lines 1 to 6). Then, as mentioned above, it is clear that when contact is not realized, the fingertip force should be equal to 0, so this is checked first (lines 7 to 8). When that is the case, $L_{\text{T-1}}$ to $L_{\text{T-N}}$ will be updated and saved to be used in the next loop (lines 9 to 12). On the other hand, if *Contact* is TRUE, the weighted average method introduced above for the saved pre-contact load cell data is used to obtain $L_{\text{challenges}}$ (line 14). Applying equations 5 (lines 16) and using the relation between L_{contact} and F_{contact} obtained from the analytics model, the estimated fingertip force F_{contact} is determined and outputted from the algorithm.

External sensor-less stable grasping technique using a virtual frame

For the three fingers to grasp object stably, in the work carried out by Tahara et al. [20], a method using only the robot state information was introduced. Then, in our previous work [29], the method was developed to be suitable for the Jinki Hand. The idea is explained as follows (see Fig. 17):

To grasp an object stably without its information using a triple-fingered robotic hand, after the three fingers are in contact with the object, an external sensor-less stable grasping technique is utilized. The technique imagines that the triangle "virtual frame" generated by connecting fingertip positions, is the object to grasp. Then, without utilizing the object properties such as size, shape, position, and posture, by exerting force for force/torque equilibrium at the equilibrium point inside the frame, the robot can grasp the object stably [20]. In [20], the controller is in a feedforward manner, and it does not utilize the contact location information between the fingertip and object. Thus, it is suitable to be applied for tactile sensor-less stable grasping with the linkage-based under-actuated hand. It is important to note here that the technique was already shown to be able to keep the grasped object stable. Here, we only apply the technique introduced in the previous work to show that the novel algorithms in this paper are effective, and the detailed explanations, including the limitations and the assumptions of the stable grasping technique, are not focused on in this paper.

In our case, since Finger 2 in the Jinki Hand cannot rotate, the "equilibrium point" was selected to be X'_{G}, which is the projection of the center of the virtual triangle X_{G} onto the Finger 2 direction:

$$\boldsymbol{X'}_{\text{G}} = \boldsymbol{X}_2 + \frac{|(\boldsymbol{X}_{\text{G}} - \boldsymbol{X}_2) \cdot \boldsymbol{F}_{\text{d\_2}}|}{|\boldsymbol{F}_{\text{d\_2}}|^2} \boldsymbol{F}_{\text{d\_2}} \tag{7}$$

in which,

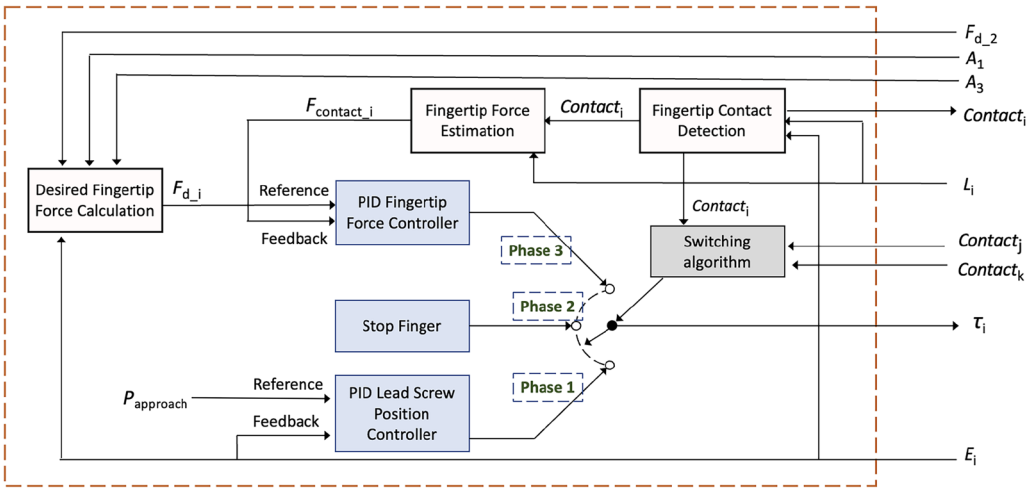


Fig. 18 Fingertip controller for each finger i

$$\boldsymbol{X}_{\mathrm{G}} = \frac{\boldsymbol{X}_1 + \boldsymbol{X}_2 + \boldsymbol{X}_3}{3} \tag{8}$$

In this equation, $\boldsymbol{X}_{\mathrm{i}}$ with i from 1 to 3 is the fingertip position of each finger i, calculated using forward kinematics on encoder data E_{i}. Thus, the controller requires the desired magnitude of the Finger 2 force $F_{\mathrm{d\_2}}$ decided by the user and the desired Finger 1 force $F_{\mathrm{d\_1}}$ and Finger 3 force $\boldsymbol{F}_{\mathrm{d\_3}}$ are automatically calculated to realize object force/torque equilibrium, as follows:

$$\begin{cases} |\boldsymbol{F}_{\mathrm{d\_1}}| = \dfrac{-F_{\mathrm{d\_2}}}{\dfrac{\sin(A_1)}{\tan(A_3)} + \cos(A_1)} \\ A_{\mathrm{d\_1}} = \langle \boldsymbol{X'}_{\mathrm{G}} - \boldsymbol{X}_1, \boldsymbol{X'}_{\mathrm{G}} - \boldsymbol{X}_2 \rangle \end{cases} \tag{9}$$

$$\begin{cases} |\boldsymbol{F}_{\mathrm{d\_3}}| = \dfrac{-F_{\mathrm{d\_2}}}{\dfrac{\sin(A_3)}{\tan(A_1)} + \cos(A_3)} \\ A_{\mathrm{d\_3}} = \langle \boldsymbol{X'}_{\mathrm{G}} - \boldsymbol{X}_3, \boldsymbol{X'}_{\mathrm{G}} - \boldsymbol{X}_2 \rangle \end{cases} \tag{10}$$

where $\langle \boldsymbol{A}, \boldsymbol{B} \rangle$ represents the angle between vectors $\boldsymbol{A}$ and $\boldsymbol{B}$. It should also be noted that the technique realizes only force/torque equilibrium in the planar plane. The vertical movement of each fingertip is limited by the degrees of freedom. Hence, the fingertip stable grasping technique relies only on the frictional force of the fingertips to realize force equilibrium in the vertical direction. In addition, the technique is modified here specifically to make it suitable for the configuration of the Jinki Hand. Since Finger 2 cannot rotate, the force/torque equilibrium that the method can support is limited when disturbance forces are applied from the opposite of Finger 2 that is not in line with Finger 2 bending direction, and are also supported passively by friction.

Stable grasp controller for the jinki hand

Fingertip controller for stable grasping

The stable grasping process starts with all three fingers approaching the object (Phase 1), each finger stops individually when contact is detected (Phase 2), then all three fingers switch to force control for stable grasping after they are all in contact with the object (Phase 3). To make this possible, each finger i will have a separate controller (see Fig. 18), and each controller will have a switch inside to change the grasping phase from 1 to 3.

Phase 1) For the fingers to approach the object in Phase 1, lead screw PID position control is used. For the desired lead screw position, the highest allowed lead screw position so the three fingers do not contact each other, P_{approach} is used, while using encoder data E as feedback to regulate the approaching process:

$$\begin{aligned} \tau_{\mathrm{finger\_i}} &= K_{\mathrm{P\_P}}(P_{\mathrm{approach}} - E_{\mathrm{i}}) \\ &+ \; ; K_{\mathrm{D\_P}} \frac{d(P_{\mathrm{approach}} - E_{\mathrm{i}})}{dT} \\ &+ K_{\mathrm{I\_P}} \int (P_{\mathrm{approach}} - E_{\mathrm{i}})\, dT \end{aligned} \tag{11}$$

Phase 2) The switch changes to Phase 2 after the Fingertip Contact Detection algorithm of the finger-

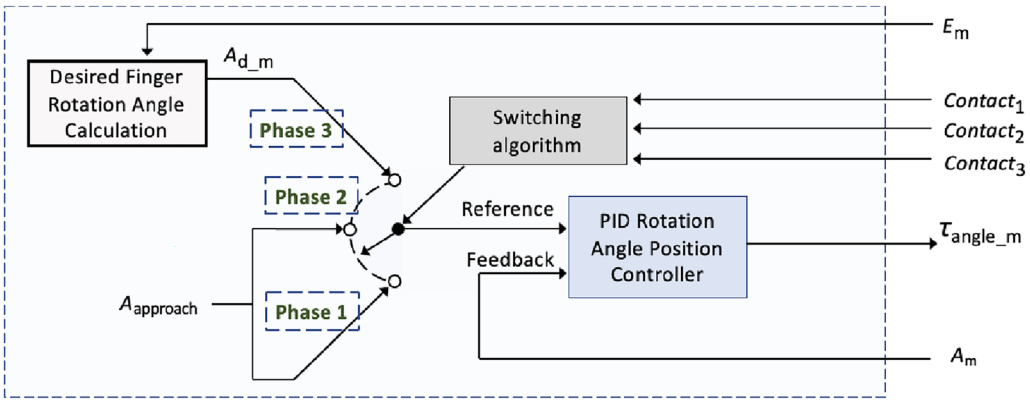


Fig. 19 Finger rotation angle controller for Fingers 1 and 3

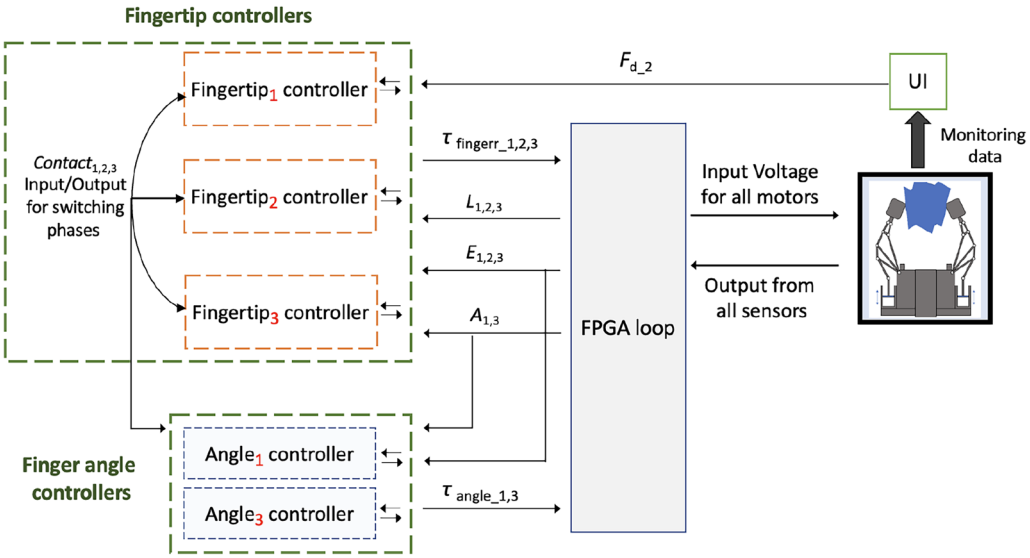


Fig. 20 Stable grasp controller for the Jinki Hand

tip controller outputs $Contact_i$ as TRUE and the signal to stop finger *i* is outputted.

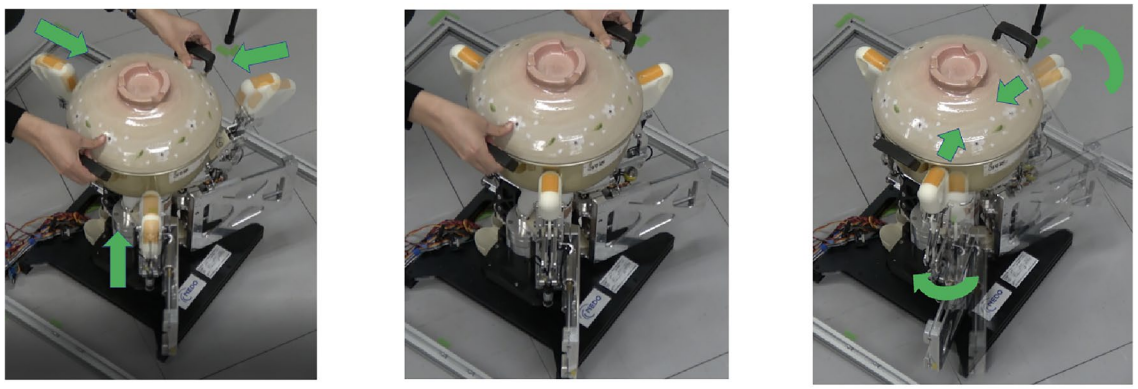
Phase 3) When $Contact_i$ of a finger and $Contact_j$ and $Contact_k$ of the other two fingers all become TRUE, the switch changes to Phase 3, where fingertip force PID control, which has F_{d_i} as the reference and $F_{contact\_i}$ as the feedback, is used:

$$\begin{aligned}\tau_{finger\_i} = {} & K_{P\_F}(F_{d\_i} - F_{contact\_i}) \\ & + K_{D\_F}\frac{d(F_{d\_i} - F_{contact\_i})}{dT} \\ & + K_{I\_F}\int (F_{d\_i} - F_{contact\_i})dT\end{aligned} \tag{12}$$

With this structure, the fingertip controller has the following input and output:

a) Input: The desired Finger 2 force F_{d_2}, and Finger 1 and 3 rotation angles A_1 A_3 to calculate the desired fingertip i force; the internal sensor load cell L and encoder E signals to be transformed into necessary information to regulate each fingertip; and the contact information of the other 2 fingers $Contact_j$ and $Contact_k$ to control the switch.

b) Output: The contact detection signal of the finger $Contact_i$ to control the switch of the other two fingers; and the torque signal τ_{finger_i} for the motor at the base of the finger to control the fingertip to the desired force.

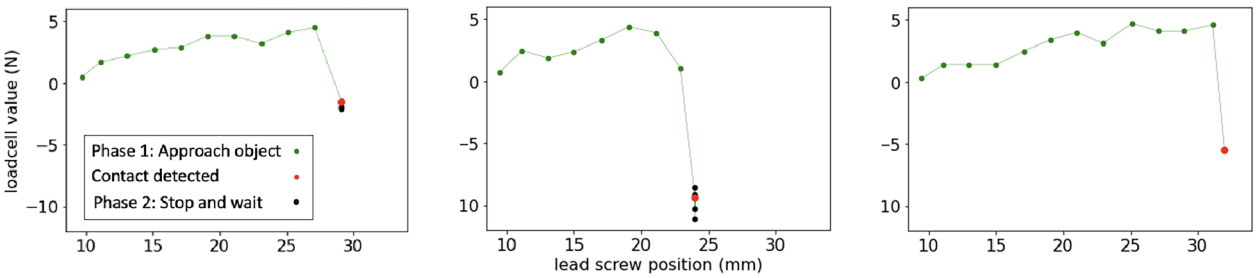


(a) Phase 1: Fingers approach the object

(b) Phase 2: Fingers stop when contact is detected

(c) Phase 3: Realizing object force/torque equilibrium

Fig. 21 Stable grasping experiment



(a) Finger 1

(b) Finger 2

(c) Finger 3

Fig. 22 Fingertip contact detection result to switch from Phase 1 to Phase 2

Finger rotation angle controller for stable grasping

The finger rotation angle controllers in Fig. 19 control the rotation angle of Finger m, where m is 1 or 3. The controller uses the same switching algorithm used in the fingertip controller. However, the algorithm is used to select the reference for the rotation angle position PID control:

$$\begin{aligned}\tau_{\text{angle\_m}} &= K_{\text{P\_A}}(A_{\text{d\_m}} - A_{\text{m}}) \\ &+ K_{\text{D\_A}}\frac{d(A_{\text{d\_m}} - A_{\text{m}})}{dT} \\ &+ K_{\text{I\_A}}\int (A_{\text{d\_m}} - A_{\text{m}})\, dT\end{aligned} \tag{13}$$

During Phases 1 and 2, the fingers approach the object to grasp. Hence, the controller reference value, $A_{\text{d\_m}}$, is the approaching angle, A_{approach}.

During Phase 3, the rotation angles of Fingers 1 and 3 are controlled to the desired angles so that they can point to the "equilibrium point" on the triangular virtual frame for object force/torque equilibrium, as explained in the "External sensor-less stable grasping technique using a virtual frame" section. To find the "equilibrium point", the encoder data E_{m} is inputted.

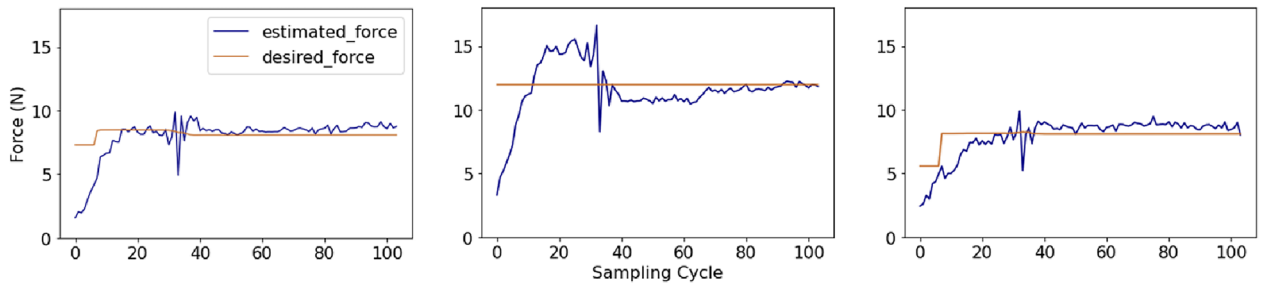
With this structure, the controller takes the lead screw position and angle position encoder data E_{m} and A_{m}, together with the contact information of all three fingers $Contact_{1,2,3}$, and outputs the signal to control the rotational angle of finger m, $\tau_{\text{angle\_m}}$

Jinki hand stable grasp controller

Fig. 20 illustrates the overall structure of the Jinki Hand controller for stable grasping.

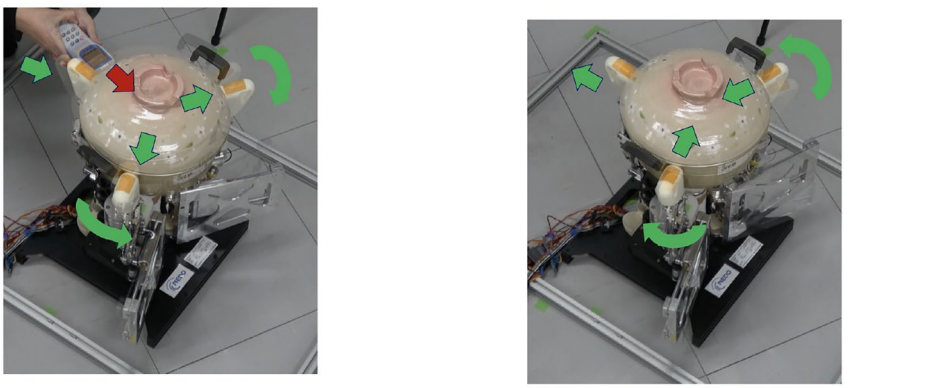
An FPGA loop is used to connect the Jinki Hand to the controllers by transforming the $\tau_{\text{finger\_1,2,3}}$ and $\tau_{\text{angle\_1,3}}$ signals into voltages to send to the robot while transforming the sensor signals into a format the controllers can use.

The fingertip controllers are connected to each other to exchange $Contact_{1,2,3}$ information, and they are also connected to the FPGA to send $\tau_{\text{finger\_1,2,3}}$ data and receive load cell $L_{1,2,3}$, encoder $E_{1,2,3}$, and angle $Angle_{1,3}$ data for feedback and desired finger force calculation. The desired



(a) Fingertip 1 (b) Fingertip 2 (c) Fingertip 3

Fig. 23 Fingertip force controller for stable grasping result during Phase 3



(a) External force is applied to the object; the fingers adjust themselves to maintain object force/torque equilibrium

(b) External force is removed; the fingers return the grasped object back close to original position

Fig. 24 Maintain object stability experiment

magnitude of Fingertip 2 force F_{d_2} is obtained from the UI, which is decided by the user. Contact information is also sent to the finger angle controllers.

The finger angle controllers receive the contact detection signals $Contact_{1,2,3}$ to control the switches inside them. Encoder data $E_{1,2,3}$ is used to obtain the desired finger angle for object force/torque equilibrium, and angle data $A_{1,3}$ is used as feedback to control the angles. These controllers output the $\tau_{angle\_1,3}$ signal to the FPGA loop.

Therefore, using this Jinki Hand controller for stable grasping, all of the necessary input/output are connected to each other.

Stable grasping experiment

The overall experiment is shown in Fig. 21. During Phase 1, the fingers approach the object from a constant angle. After the algorithm detects contact between a fingertip and the object, the controller moves to Phase 2 to stop the finger movement until all three fingers get in contact with the object. In Phase 3, the fingers move to the desired angle and exert the desired fingertip force to realize object force/torque equilibrium.

In [29], the data from the load cells, encoders, estimated force, and desired force are collected. In this study, the collected data is further analyzed to evaluate the Fingertip Contact Detection and Fingertip Force Estimation algorithms.

During the experiment, the load cell values $L_{1,2,3}$ and $E_{1,2,3}$, together with the $Contact_{1,2,3}$ data were collected. By plotting these values, the result of the Fingertip Contact Detection algorithm are shown (see Fig. 22). In the graph, the green points are the real-time data, the red point shows the moment contact was detected, and the back points represent the post-contact points. As seen in the three graphs, contact was detected right after the data point deviated significantly from the non-contact behavior. The controller switch changed the grasping phase to Phase 2, and each finger stopped moving right after contact. Therefore, the monitored data showed that the contact detection algorithm worked well for stopping the fingers when they were in contact during the approaching phase.

The desired fingertip force F_{d_i} and estimated fingertip force $F_{contact\_i}$ were also collected during the experiment (see Fig. 23). They confirmed that the estimated fingertips force reached the desired values for object force/torque equilibrium. For that reason, from this graph, it is confirmed that the controller gain parameters $K_{P,I,D}$ are tuned well. Therefore, whether the robotic hand can grasp an object stably depends mainly on the quality of the Fingertip Force Estimation algorithm.

To evaluate whether the proposed Fingertip Force Estimation algorithm is suitable for stable grasping, the quality of stable grasping is evaluated. As explained above, due to the vertical movement of each fingertip being limited by the degrees of freedom, the control technique relies only on the frictional force to maintain equilibrium in the vertical direction. In addition, the adaptability to the opposite side of Finger 2 is limited because it cannot rotate freely. Thus, in order to verify that the stable grasping is established by the controller, it is necessary to apply a force in the horizontal plane from the opposite side of Finger 1 and 3. To do this, after the object was grasped and the fingers realized object force/torque equilibrium, an external force of 10 to 15 N was applied to the grasped object (see Fig. 24). During that process, the fingers automatically adjusted themselves to maintain object force/torque equilibrium and kept the object in the hand without falling. This includes adjusting the fingertip forces and the finger angles to satisfy the new desired conditions for object force/torque equilibrium. After the external force was removed from the grasped object, the fingers returned the object to near the original position before the object was pushed. Therefore, this experiment showed that the Fingertip Force Estimation algorithm is accurate enough to regulate the fingertip forces to keep an object stable. As a result, the controller can be further developed to be used for dexterous manipulation.

Conclusion

In this study, we propose two data-driven-based algorithms: the Fingertip Contact Detection algorithm and the Fingertip Force Estimation algorithm, which are designed to use only internal sensor data to operate in real-time within a stable grasp controller for a triple-fingered linkage-based under-actuated robotic hand. Using these algorithm, the purpose of this study was to automate the process of approaching an object, grasping, realizing object force/torque equilibrium, and maintaining that state.

The Fingertip Contact Detection algorithm uses the traditional DBSCAN algorithm to "learn" (tune parameters) non-contact data point characteristics collected beforehand. Using the learned information, a density-based-clustering approach is introduced to detect whether the internal sensors data are contact or non-contact data points in real-time. This method assumes that the collected dataset can represent the possible behavior of the internal sensor data in the approaching phase. The experiment results conclude that the method can perform this task well and can also be adjusted to be used for other grasping/manipulation tasks.

The Fingertip Force Estimation algorithm uses the contact detection output and weight average method to measure challenging parts to model in real-time. Using this method, the analytics model assumptions and the sensor noise are significantly reduced, which improves the controller's accuracy. When using this method, it is assumed that the finger posture does not change much during Phases 2 and 3, hence $L_{challenges}$ remains constant. The experiment confirmed that this algorithm can obtain enough accurate fingertip force data to regulate the fingertip force in real-time to realize and maintain object force/torque equilibrium, and the assumption used is appropriate.

In summary, by replacing the parts which are difficult to model in the model-based stable grasp controller with data-driven-based algorithms, this paper proposes a stable grasp controller for a triple-fingered linkage-based under-actuated robotic hand. The proposed controller was tested to be able to maintain object force/torque equilibrium, and it can now be used in the future as the foundation of other in-hand manipulation controllers.

Acknowledgements
A part of this work were supported by the Cabinet Office (CAO), Cross-ministerial Strategic Innovation Promotion Program (SIP), "An intelligent knowledge processing infrastructure, integrating physical and virtual domains" (funding agency: NEDO), and JSPS KAKENHI Grant Number 20H00610.

Author contributions
All the contents included in the study, such as the idea, method, theory, simulation, and experiments were created and performed by the authors. All authors read and approved the final manuscript.

Funding
A part of this work were supported by the Cabinet Office (CAO), Cross-ministerial Strategic Innovation Promotion Program (SIP), "An intelligent knowledge processing infrastructure, integrating physical and virtual domains" (funding agency: NEDO), and JSPS KAKENHI Grant Number 20H00610.

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 15 May 2023 Accepted: 21 March 2024
Published online: 05 April 2024

References

1. Aukes D, Heyneman B, Ulmen J, Stuart H, Cutkosky M, Kim S, Garcia P, Edsinger A (2014) Design and testing of a selectively compliant underactuated hand. Int J Robot Res 33:721–735. https://doi.org/10.1177/0278364913518997
2. Odhner LU, Jentoft LP, Claffee MR, Corson N, Tenzer Y, Ma RR, Buehler M, Kohout R, Howe RD, Dollar AM (2014) A compliant, underactuated hand for robust manipulation. Int J Robot Res 33(5):736–752. https://doi.org/10.1177/0278364913514466
3. Ma RR, Odhner LU, Dollar AM (2013) A modular, open-source 3d printed underactuated hand. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 2737–2743. https://doi.org/10.1109/ICRA.2013.6630954
4. Deimel R, Brock O (2016) A novel type of compliant and underactuated robotic hand for dexterous grasping. Int J Robot Res 35(1–3):161–185. https://doi.org/10.1177/0278364915592961
5. He B, Wang S, Liu Y (2019) Underactuated robotics: A review. Int J Adv Robot Syst. https://doi.org/10.1177/1729881419862164
6. Kinugawa J, Suzuki H, Terayama J, Kosuge K (2022) Underactuated robotic hand for a fully automatic dishwasher based on grasp stability analysis*. Adv Robot 36(4):167–181. https://doi.org/10.1080/01691864.2021.2011778
7. Yoon D, Choi Y (2017) Underactuated finger mechanism using contractible slider-cranks and stackable four-bar linkages. IEEE/ASME Trans Mechatron 22(5):2046–2057. https://doi.org/10.1109/TMECH.2017.2723718
8. Wu L, Carbone G, Ceccarelli M (2009) Designing an underactuated mechanism for a 1 active dof finger operation. Mech Mach Theory 44(2):336–348. https://doi.org/10.1016/j.mechmachtheory.2008.03.011
9. Odhner LU, Dollar AM (2015) Stable, open-loop precision manipulation with underactuated hands. Int J Robot Res 34(11):1347–1360. https://doi.org/10.1177/0278364914558494
10. Su Y, Wu Y, Lee K, Du Z, Demiris Y (2012) Robust grasping for an underactuated anthropomorphic hand under object position uncertainty. In: 2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 719–725. https://doi.org/10.1109/HUMANOIDS.2012.6651599
11. Fan S, Gu H, Zhang Y, Jin M, Liu H (2018) Research on adaptive grasping with object pose uncertainty by multi-fingered robot hand. Int J Adv Robot Syst 15(2):1729881418766783. https://doi.org/10.1177/1729881418766783
12. Choi S-H, Tahara K (2020) Dexterous object manipulation by a multi-fingered robotic hand with visual-tactile fingertip sensors. ROBOMECH J 7(1):14. https://doi.org/10.1186/s40648-020-00162-5
13. van Hoof H, Hermans T, Neumann G, Peters J (2015) Learning robot in-hand manipulation with tactile features. 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), 121–127. https://doi.org/10.1109/HUMANOIDS.2015.7363524
14. Ha V, Ha C, Dang Khoa N (2016) A general contact force analysis of an under-actuated finger in robot hand grasping. Int J Adv Robot Syst 13:1. https://doi.org/10.5772/62131
15. Andrychowicz OM, Baker B, Chociej M, Józefowicz R, McGrew B, Pachocki J, Petron A, Plappert M, Powell G, Ray A, Schneider J, Sidor S, Tobin J, Welinder P, Weng L, Zaremba W (2020) Learning dexterous in-hand manipulation. Int J Robot Res 39(1):3–20. https://doi.org/10.1177/0278364919887447
16. Kappassov Z, Corrales J-A, Perdereau V (2015) Tactile sensing in dexterous robot hands – review. Robot Auton Syst 74:195–220. https://doi.org/10.1016/j.robot.2015.07.015
17. Arimoto S (2008) Control theory of multi-fingered hands: a modelling and analytical-mechanics approach for dexterity and intelligence. Control Theory Multi-Fingered Hands. https://doi.org/10.1007/978-1-84800-063-6
18. Arimoto S, Nguyen PTA, Han H-Y, Doulgeri Z (2000) Dynamics and control of a set of dual fingers with soft tips. Robotica 18(1):71–80. https://doi.org/10.1017/S0263574799002441
19. Arimoto S (2007) A differential-geometric approach for 2d and 3d object grasping and manipulation. Ann Rev Control 31(2):189–209. https://doi.org/10.1016/j.arcontrol.2007.08.004
20. Tahara K, Arimoto S, Yoshida M (2009) Dynamic force/torque equilibrium for stable grasping by a triple robotic fingers system. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2257–2263. https://doi.org/10.1109/IROS.2009.5354563
21. Suehiro K, Ozawa R, Van Heerden K (2017) Force-sensorless fault tolerant detection and switching control of tendon-driven mechanisms with redundant tendons. IEEE Robot Autom Lett 2(4):2248–2254. https://doi.org/10.1109/LRA.2017.2724771
22. Liarokapis MV, Dollar AM (2016) Learning task-specific models for dexterous, in-hand manipulation with simple, adaptive robot hands. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2534–2541. https://doi.org/10.1109/IROS.2016.7759394
23. Liarokapis M, Dollar AM (2017) Deriving dexterous, in-hand manipulation primitives for adaptive robot hands. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1951–1958. https://doi.org/10.1109/IROS.2017.8206014
24. Liarokapis M, Dollar AM (2019) Combining analytical modeling and learning to simplify dexterous manipulation with adaptive robot hands. IEEE Trans Autom Sci Eng 16(3):1361–1372. https://doi.org/10.1109/TASE.2018.2885801
25. Silver T, Allen K, Tenenbaum J, Kaelbling L (2018) Residual policy learning. arXiv Preprint. https://doi.org/10.48550/arXiv.1812.06298
26. Johannink T, Bahl S, Nair A, Luo J, Kumar A, Loskyll M, Ojea JA, Solowjow E, Levine S (2019) Residual reinforcement learning for robot control. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 6023–6029. IEEE
27. Doan HTL, Tahara K (2022) Fingertip contact detection for a multi-fingered under-actuated robotic hand using density-based clustering method. In: The Robotics and Mechatronics Conference 2022 in Sapporo (Robomech)
28. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. KDD'96, pp. 226–231. https://doi.org/10.5555/3001460.3001507. AAAI
29. Doan HTL, Tahara K (2023) Data-driven-based stable object grasping for a triple-fingered under-actuated robotic hand. In: 2023 IEEE/SICE International Symposium on System Integrations (SII)
30. Man-Machine Synergy Effectors, Inc. https://www.jinki.jp/

Publisher's Note