


Article

Behavior Cloning and Replay of Humanoid Robot via a Depth Camera

Quantao Wang ¹, Ziming He ¹, Jialiing Zou ², Haobin Shi ^{1,*} and Kao-Shing Hwang ³ ¹ School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China² School of Software, Northwestern Polytechnical University, Xi'an 710129, China³ Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan

* Correspondence: shihaobin@nwpu.edu.cn

Abstract: The technique of behavior cloning is to equip a robot with the capability of learning control skills through observation, which can naturally perform human–robot interaction. Despite many related studies in the context of humanoid robot behavior cloning, the problems of the unnecessary recording of similar actions and more efficient storage forms than recording actions by joint angles or motor counts are still worth discussing. To reduce the storage burden on robots, we implemented an end-to-end humanoid robot behavior cloning system, which consists of three modules, namely action emulation, action memorization, and action replay. With the help of traditional machine learning methods, the system can avoid recording similar actions while storing actions in a more efficient form. A jitter problem in the action replay is also handled. In our system, an action is defined as a sequence of many pose frames. We propose a revised key-pose detection algorithm to keep minimal poses of each action to minimize storage consumption. Subsequently, a clustering algorithm for key poses is implemented to save each action in the form of identifiers series. Finally, a similarity equation is proposed to avoid the unnecessary storage of similar actions, in which the similarity evaluation of actions is defined as an LCS problem. Experiments on different actions have shown that our system greatly reduces the storage burden of the robot while ensuring that the errors are within acceptable limits. The average error of the revised key-pose detection algorithm is reduced by 69% compared to the original and 26% compared to another advanced algorithm. The storage consumption of actions is reduced by 97% eventually. Experimental results demonstrate that the system can efficiently memorize actions to complete behavioral cloning.

Keywords: behavior cloning; humanoid robot; action emulation; action representation; action combination; key-pose

MSC: 68T05

Citation: Wang, Q.; He, Z.; Zou, J.; Shi, H.; Hwang, K.-S. Behavior Cloning and Replay of Humanoid Robot via a Depth Camera. *Mathematics* **2023**, *11*, 678. <https://doi.org/10.3390/math11030678>

Academic Editors: Florin Leon and Ivan Lorencin

Received: 27 December 2022

Revised: 16 January 2023

Accepted: 25 January 2023

Published: 29 January 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Nowadays, the role of robots is highlighted due to their great potential, popularity [1] and the social influence [2]. Human–computer interaction is always the focus of robotics research, where safety [3], personalization [4], output complexity [5], and data collection [6] should all be considered. To improve the social identity of human–computer interaction, the primary problem to be solved is adapting the robot to the characteristics of human motion. The action and posture of the humanoid robot are essential manifestations of the robot's ability, which will also have a profound impact on the user. Therefore, how to control the robot's limb movement is a challenging task, and it is also an essential problem in robot research. In this work, human–computer interaction is implemented through behavioral cloning. Behavioral cloning is the most direct human–computer interaction [7]. In behavioral cloning, a human trainer first demonstrates the control task by observing a visualization of sensor data that is available to an autonomous controller and selecting the

appropriate action that the controller should perform. Scholars from various countries have conducted research on this topic. Ref. [8] proposed the concept and method of basic motion set that is used to establish the basic motion set of the dynamic system, and study the motion learning problem of the humanoid robot. Refs. [9,10] showed how to evaluate the robots' memory of pose. Ref. [11] achieved human-style locomotion with reinforcement learning methods, Ref. [12] used meta-learning to implement behavioral cloning, Ref. [13] designed neural networks for particular joints to enhance one-to-one mapping imitation. In addition, and Ref. [14] created a robot simulation learning platform that uses a series of evaluation indicators to evaluate the accuracy of robot motion simulation. While some achievements have been made in behavioral cloning, there are still aspects to be improved. First, due to the presence of environmental noise, the sensor has real-time variability when acquiring human joint data, which makes the robot easily cause jitter in action replay. Second, the same or similar actions performed by human demonstrators are often unnecessarily saved by robots. Third, data of partially similar actions are always stored separately in the form of the angles or motor counts of joints, consuming a large storage space. What is more, when the humanoid robot has limited computational performance, it is difficult to implement larger neural networks that allow it to autonomously perform saved actions.

In this work, a key-pose detection algorithm was proposed to keep minimal poses of each action. When the RGB-D camera sends the action data, the value of the rotation speed of the joints and the rotation direction of the joints on the robot will be used to determine whether the corresponding pose is a key pose. The value of each key-pose is assigned by the dissimilarity formula to distinguish them. In addition, we propose to implement a clustering algorithm [15], which documents the data by assigning each pose to its suitable group, and each group is labeled with a different ID. It not only effectively improves the memory efficiency of actions, but also further calculates the similarity between actions.

The remainder of this paper is organized as follows: Section 2 introduces two algorithms that we proposed and the functions of the designed system on the humanoid robot. The implementation of the system and the experimental data are presented in Section 3. In Section 4, the conclusion and future work are given.

2. Our Works

The architecture of the proposed system is shown in Figure 1. The personal computer (PC) in the system implements an inverse-kinematic algorithm and interpolation algorithm to process the data collected by the Kinect, a line of motion sensing input devices, and sends the results to the robot via a wireless network. On the humanoid robot side, three basic functional modules are implemented. Respectively, they serve for action emulation, action replay, and action memorization. When the RGB-D camera observes an action taken by a particular user, it sends the raw data to the PC, which further processes the data to convert them into geometrical information that is compatible with the modules on the robot.

2.1. Key-Pose Detection Algorithm

'Key pose' is a term that originates in the animation field, where an action can be fully reflected by several key poses extracted from the action. It has already been applied in academic research as well. In our system, an action is defined as a sequence of many pose frames. Key poses are defined as the essential poses taken out of the sequence of poses of an action and are capable of composing a new sequence of poses necessary to recreate the action, from which 'essential' means that once one of the key poses is deleted, the action created by the new sequence of key poses will have an unacceptable error value compared to the original action, or will not look similar to the original action at all.

When implementing behavioral cloning on robots, if image frames collected by the RGB-D camera are directly recorded to represent any actions, a large amount of storage space will be consumed. Additionally, it costs huge calculating resources to process and retrieve image data, which may result in lags in the system. Therefore, the key-pose detection algorithm is proposed to identify and record the key poses.

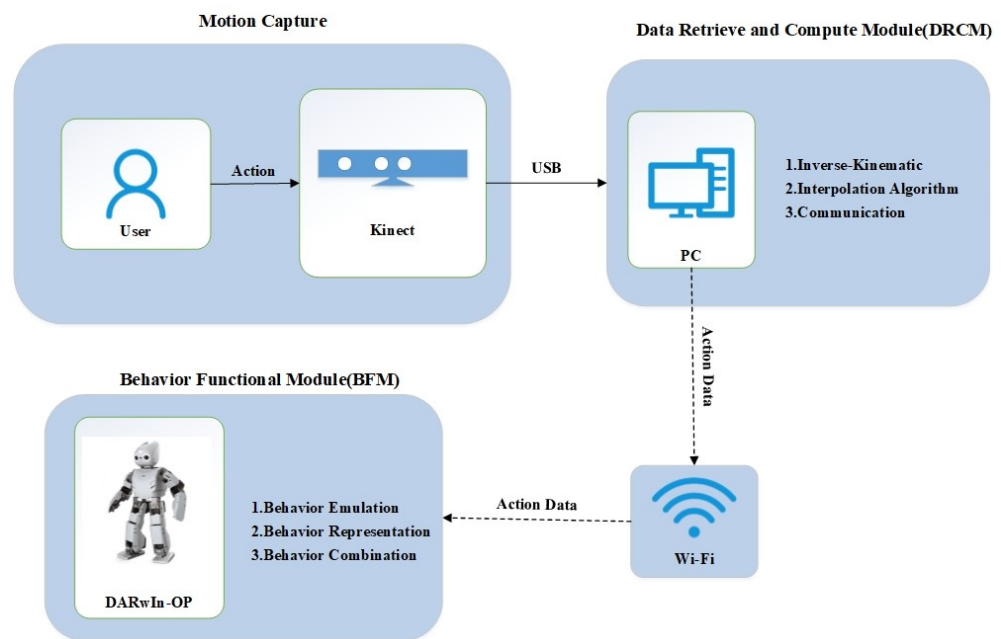


Figure 1. System architecture.

UDP, user datagram protocol, is chosen as the communication protocol in our system since retrieving data from the RGB-D camera only costs about 70 ms each time. Because of the relatively short time intervals, the action emulation can work even if some packets are lost. As is shown in Figure 2, each UDP packet contains motor counts, a bit for conforming key posture, a bit for conforming continuity, the group ID of the key pose, and the time interval needed by the pose.

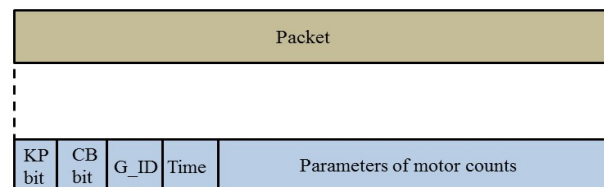


Figure 2. Architecture of packet.

'KP_bit', 'CB_bit', 'G_ID', and 'Time' in Figure 2 respectively refer to the 'key pose bit' (the bit for conforming key posture), 'continuous bit' (the bit for conforming continuity), the group ID of the key pose and the time interval between the next key pose and the key pose. If the value of 'KP_Bit' is 0, the value of 'Time' will be 0 as well. Otherwise, if the value of 'KP_Bit' is not 0, the value of 'Time' will be exactly the time interval between two key poses.

Algorithm 1 shows how each key pose is identified. Each time the skeleton data are generated, the data are simultaneously determined whether they are a key pose or not by the information about the rotation direction and speed of joints in the skeleton data. In Algorithm 1, two variables recording current and previous states are set to check whether the state in the algorithm changes. Firstly, inverse kinematics is implemented to calculate the rotation angle of each joint. Then, the value of the current angles of the joints is compared with the previous value. The state that the discrepancy between two angles is bigger than a preset threshold is defined as an 'action state'. Otherwise, it is a 'static state'. If the current state differs from the previous, key_pose_bit will be set as 1; otherwise, it is set as 0. After that, if the key_pose_bit is 0, the rotation directions of each joint are checked to be changed or not. If they are changed, the key_pose_bit will be set as 1. Otherwise, it will be designated as 0. All the data in the algorithm will be saved.

In this paper, we modified formula (1) into formula (2) to calculate the dissimilarity of poses so that the errors will be reduced:

$$\rho_{i,i+1} = \left| \frac{(n-1)\sigma_i\sigma_{i+1} - \sum_{j=1}^{12} (\beta_{i,j} - \bar{\beta}_j)(\beta_{i+1,j} - \bar{\beta}_j)}{(n-1)\sigma_i\sigma_{i+1}} \right| \tag{1}$$

where $\rho_{i,i+1}$ presents the calculated dissimilarity value, n is the total number of poses, σ is the standard deviation of pose numbered i , $\beta_{i,j}$ presents the angle of the joint j in the pose numbered i , and $\bar{\beta}_j$ is the average value of different angles of the joint numbered j in all poses:

$$\rho_{i-1,i,i+1} = 1 - \frac{\sum_{j=1}^N (|\Delta\phi_{i-1,i,j} - \overline{\Delta\phi_j}| \cdot |\Delta\phi_{i,i+1,j} - \overline{\Delta\phi_j}|)}{N(\sigma_{i-1,i})(\sigma_{i,i+1})} \tag{2}$$

where $\rho_{i-1,i,i+1}$ presents the calculated dissimilarity value, N is the total number of joints, $\sigma_{i-1,i}$ is the standard deviation of changes in values of joints angles of poses numbered i and $i - 1$. $\Delta\phi_{i-1,i,j}$ is the angle change of poses numbered i and $i - 1$ in joint j , and $\overline{\Delta\phi_j}$ represents the average of all the variation in joint numbered j of the poses:

Algorithm 1 Key-pose detection algorithm.

```

1: while not Terminated do
2:   Calculate Dissimilarity Between Previous Data Via Inverse-Kinematics
3:   if The Dissimilarity ≥ Threshold then
4:     CurrentState = ActionState
5:   else
6:     CurrentState = StaticState
7:   end if
8:   if CurrentState == PreState then
9:     key_pose_bit = 1
10:  else
11:    if the Direction is Changed then
12:      key_pose_bit = 1
13:    else
14:      key_pose_bit = 0
15:    end if
16:  end if
17:  Save Data
18: end while

```

2.2. Clustering Algorithm

The concept of grouping is always used in data mining and many other fields. In our work, grouping is defined as similar to clustering in data mining. With a key-pose detection algorithm, raw data from the RGB-D camera are transformed into data symbolizing essential poses, which are stored by way of motor counts and are less storage-consuming. In this paper, the agglomerative hierarchical clustering algorithm [15] is applied to the data of key poses to document them. Each pose will be assigned to a group with the group ID, and in this way, the storage load is reduced, and the similarity between actions can be further calculated with such group IDs. Figure 3 illustrates the changes in groups through the clustering algorithm, where each red dot presents the central point in the group it belongs to. Additionally, due to the geometric position of each red dot, every such dot can represent all the data points in the same group.

The key poses will be divided into several groups after the clustering algorithm, and each group will be assigned a unique group ID. Instead of recording motor counts, each action is stored by recording its group ID. Table 1 shows the storage space status of the humanoid robot. Table 2 compares the storage consumption of saving group ID and the

storage consumption of saving motor counts. As Table 2 shows, saving each action by group ID reduces the total storage consumption substantially. In total, 450,000 actions could be saved in the format of group ID, which is over 30 times more than the maximum number of actions saved by recording the motor counts, which is only 13,000, as is shown in Table 2.

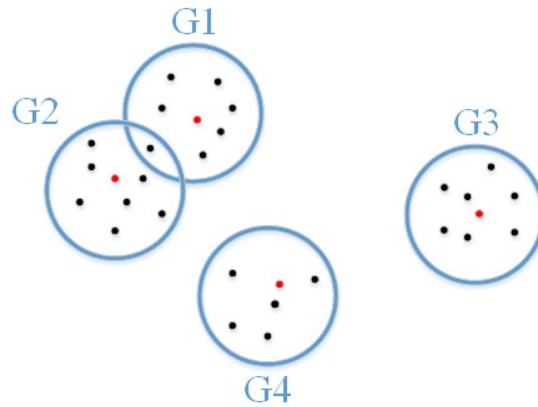


Figure 3. Illustration of group.

Table 1. Storage space of the humanoid robot.

Storage Space	Size
Total	3.6 GB
Used	2.3 GB
Available	1.3 GB

Table 2. Different format comparison.

Item	Group ID	Motor Counts
Action Size	About 3 KB	About 100 KB
Number of Action	About 450,000	About 13,000

2.3. The Humanoid Robot Functions

2.3.1. Action Emulation

The humanoid robot retrieves the built-in library to find and calls for matched data when it receives processed data from the PC via Wi-Fi to synchronously imitate the user’s actions. Due to the discrepancy between the DOF, degree of freedom, of the humanoid robots’ shoulders, which is 2, and the DOF of the human shoulder, which is 3, humanoid robots cannot achieve complete imitation. In addition, synchronous imitation on the lower body is impossible to be reached due to the following two reasons:

- RGB-D camera has low stability on the lower body skeleton relatively.
- Robots have no ability to balance themselves.

Because of the above two reasons, only upper motors can be applied to synchronous imitation. When imitating lower body poses, some robots’ actions may differ from human actions. For instance, it is hard to find a way to process the raw data generated by the RGB-D camera focusing on a standing user to obtain precise knee joint data. Even if the knee joint data are accurate, chances are that the humanoid robot will fall backward or forward when performing emulation in the lower body due to imbalance.

The humanoid robot unpacks the UDP packets after receiving them from the PC. Then, it calls the functions in its local library and runs the emulation that keeps running until the end of the program.

2.3.2. Action Memorization

In our system, an action is defined as a sequence of many pose frames. As mentioned in the previous paragraph, each pose is assigned a unique group ID after the clustering algorithm. Therefore, each action could consist of a series of group IDs. The action can be compared with the groups' IDs it concerns this way. To further reduce the storage cost, we propose a method to determine whether the latest received action data are worth being saved locally in the database. When the humanoid robot receives the action data, it checks whether the same data exist in the local database. If the data already exist, the data will be ignored.

When comparing two actions, the discrepancy in the key poses that they could be split into and the discrepancy in the speed of shifting the key poses from one to the other are noted. In our system, discrepancies in speed between actions are ignored when calculating the similarity between two actions because of the high storage cost of recording all the time variables. However, speed differences between actions will still be active when implementing the system. When calculating the similarity of actions, they are compared by the group IDs of the key poses contained by them. Evaluating the similarity of two series of key poses of two actions is characterized as solving the LCS, longest common subsequence, problem in our system. The longest common subsequence (LCS) problem [16] is the problem of finding the longest subsequence common to all sequences in a set of sequences. A similarity table is established to restore all the similarities. Algorithm 2 is the modified LCS algorithm procedure. The number of the longest common subsequence between two actions will be figured out by the algorithm, and Equation (3) shows how to calculate the similarity rate with the results of the LCS algorithm:

$$\text{Similarity Rate} = \frac{N(\text{LCS}(A_1, A_2))}{n} \quad (3)$$

where *Similarity Rate* presents how similar the two actions are. *n* is the number of segments of the compared actions; A_1, A_2 respectively represent the data of action numbered 1 and 2; and $N(\text{LCS}(A_1, A_2))$ represents the number of LCS.

2.3.3. Action Replay

The robot has to achieve consistency in the representation and speed of the user's actions. Such consistency can be attained with the help of applying and clustering algorithms, which will minimize the error between the representation and the original action. Figure 4 illustrates how to calculate the speed of each action. In Figure 4, 'T' refers to the time consumed. Each elapsed time interval between two key poses is calculated and can be further used to compute the changes in the speed of every joint. As shown in Figure 5, the elapsed time between the Key Pose labeled 1 and the Key Pose labeled 2 is (T1+T2+T3). The speed of the whole action can be calculated with the data from the motor or the number of total key poses.

Algorithm 2 Revised LCS algorithm.

```

1: function LCS( $S_1, S_2$ )
2:   for  $i = 0, 1, \dots, m$  do
3:      $len(i, 0) = 0$ 
4:   end for
5:   for  $j = 1, 2, \dots, n$  do
6:      $len(0, j) = 0$ 
7:   end for
8:   for  $i = 1, 2, \dots, m$  do
9:     for  $j = 1, 2, \dots, n$  do
10:      Search similarity table
11:      if  $a_i = b_j$  then
12:         $len(i, j) = len(i-1, j-1) + 1$ 
13:      end if
14:      if  $len(i-1, j) \geq len(i, j-1)$  then
15:         $len(i, j) = len(i-1, j)$ 
16:      else
17:         $len(i, j) = len(i, j-1)$ 
18:      end if
19:    end for
20:  end for
21: end function
    
```

As is shown in Figure 4, after taking data consistency and speed variations into account, there still exist standstills in the progress of imitation because the humanoid robot will directly implement the action data without any optimizing processing. Thus the whole action played by the robot seems to be not natural.

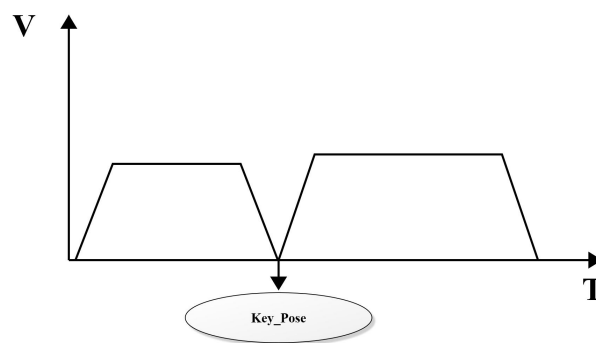


Figure 4. Continuous motion interrupt diagram.

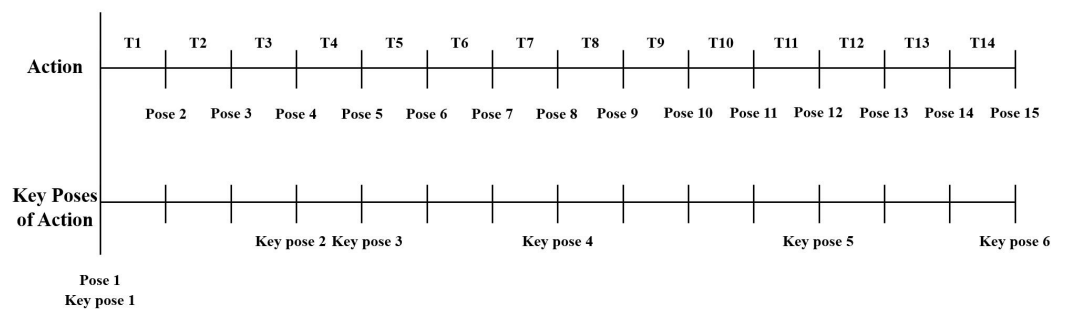


Figure 5. Relation of poses and key poses.

To solve the problem, a ‘continuous bit’ is added to the program. It is proposed to symbolize whether the state of the robot is active or not. The continuous bit is either 0 or 1. In the program, the speed of each joint is checked. If it is stable, implying that all the

velocities of the joints are 0, the continuous bit will be set as 0, meaning that the robot is in a static state; otherwise, if there still exists any joint that is constantly moving, the continuous bit will be set as 1, meaning that the robot is in an active state, as is shown in Figure 6.

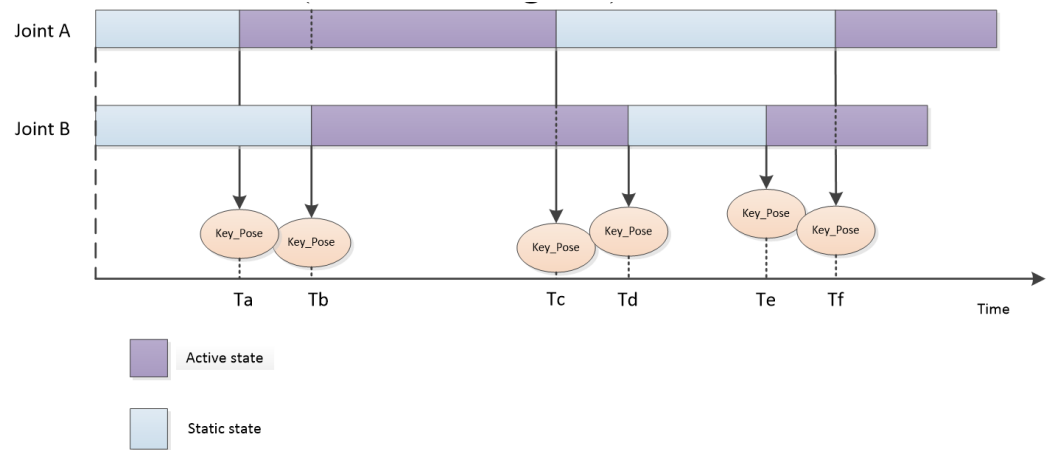


Figure 6. Influence of key-pose detection diagram.

The continuous bit is used to enable the robot to perform the actions smoothly. If the current continuous bit is 1, the number of each motor count will add or remove a constant so that the motor will keep its speed and direction for a while after reaching its original goal position. The current motor count is continuously checked to ensure the accuracy of the action. The actual goal position of each pose may be changed because of the functions of the continuous bit. However, once the check and a particular pose are done, the motors on the robot will return to the original goal position or move to the next goal position. In this way, each motor will not halt when the robot reads the next key pose while the accuracy is ensured, as shown in Figure 7.

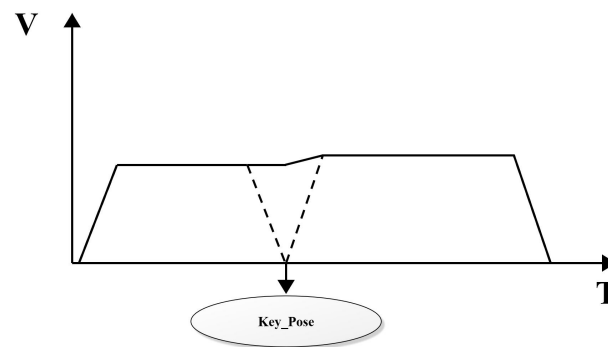


Figure 7. Improved of continuous motion diagram.

3. Implementation

3.1. Implementation of the Key-Pose Detection Algorithm

In this section, the method to identify key poses will be introduced. Two other methods are to be discussed besides the algorithm, which is revised from the key-pose detection algorithm based on dissimilarity, adopted in our study. They are the condition key-pose detection algorithm [17] and key-pose detection algorithm based on dissimilarity [18].

To evaluate the proposed key-pose detection algorithm, several experiments are conducted. Figure 8a is labeled Action I in the experiment; Figure 8b, Action II; Figure 8c, Action III, Figure 8d, Action IV. Action I, Action II, Action III and Action IV explain the research results.

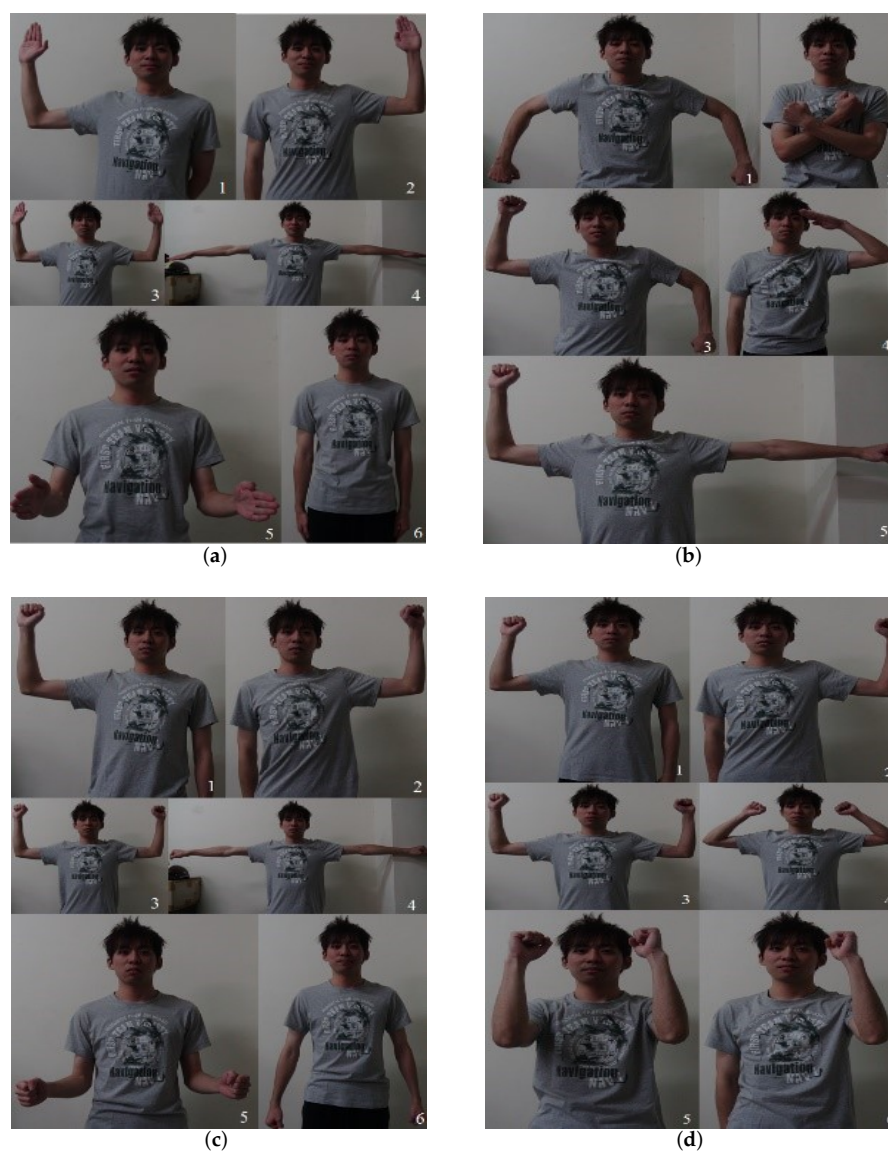


Figure 8. Action I (a), Action II (b), Action III (c) and Action IV (d).

To ensure accuracy, each action consists of 250 poses, taking nearly 15 seconds to be executed. Additionally, six joints of the humanoid robot are set to active to perform the action. They are respectively Left_Elbow, Left_Shoulder_Roll, Left_Shoulder_Pitch, Right_Elbow, Right_Shoulder_Roll, Right_Shoulder_Pitch.

After the action (Action I) performed by the user is received and processed by the RGB-D camera, three different algorithms will further process the data. The three corresponding results are shown below. Figures 9–11 show the comparison between these three algorithms. Figure 9 presents the result for the condition key-pose detection algorithm. Figure 10 shows the result for the key-pose detection algorithm based on dissimilarity. Figure 11 is the result for the key-pose detection algorithm revised from the key-pose detection algorithm based on dissimilarity, which is used in our implementation.

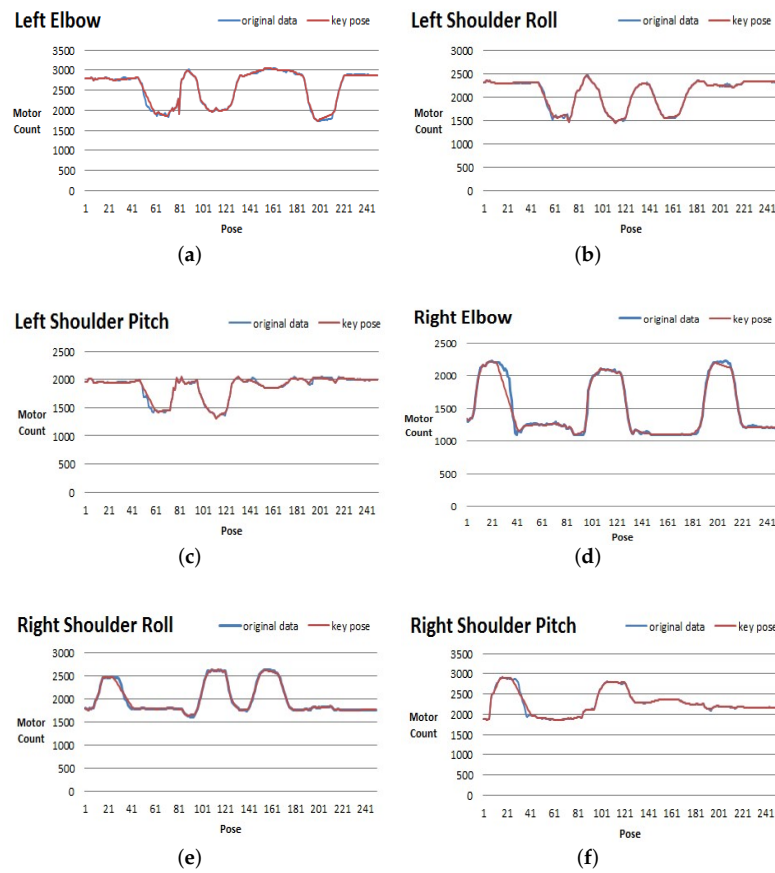


Figure 9. Results of [17] condition key-pose detection algorithm versus original pose data: (a) motor counts data on Left Elbow. (b) Motor counts data on Left Shoulder Roll. (c) Motor counts data on Left Shoulder Pitch. (d) Motor counts data on Right Elbow. (e) Motor counts data on Right Shoulder Roll. (f) Motor counts data on Right Shoulder Pitch.

Table 3 compares the average error values of the three algorithms with the same amount of key poses. As is shown in Table 3, after the modification, the key-pose detection algorithm based on dissimilarity can achieve the lowest error value, reducing 69% error compared to the original, 26% compared to the condition key-pose detection algorithm. The error is defined as the average value of the difference between the angles of each joint of the original poses and the key poses, as Equation (4) shows:

$$Error = \frac{\sum_{j=1}^6 \sum_{n=1}^{250} |K_n - O_n|}{6 \times 250} \times 0.0878 \tag{4}$$

where j is the label of the joint, n is the label of the data of the pose, K_n is the angles value of joints in the key pose, and O_n is the angles value of the joints in the original pose.

Table 3. Comparison of the three key-pose detection algorithms.

Algorithm	Average of Error	Amount of Key Poses
Condition Key-Pose Algorithm	1.24146	250 → 100
Original Dissimilarity Key-Pose Algorithm	2.74349	250 → 100
Revised Dissimilarity Key-Pose Algorithm	0.91062	250 → 100

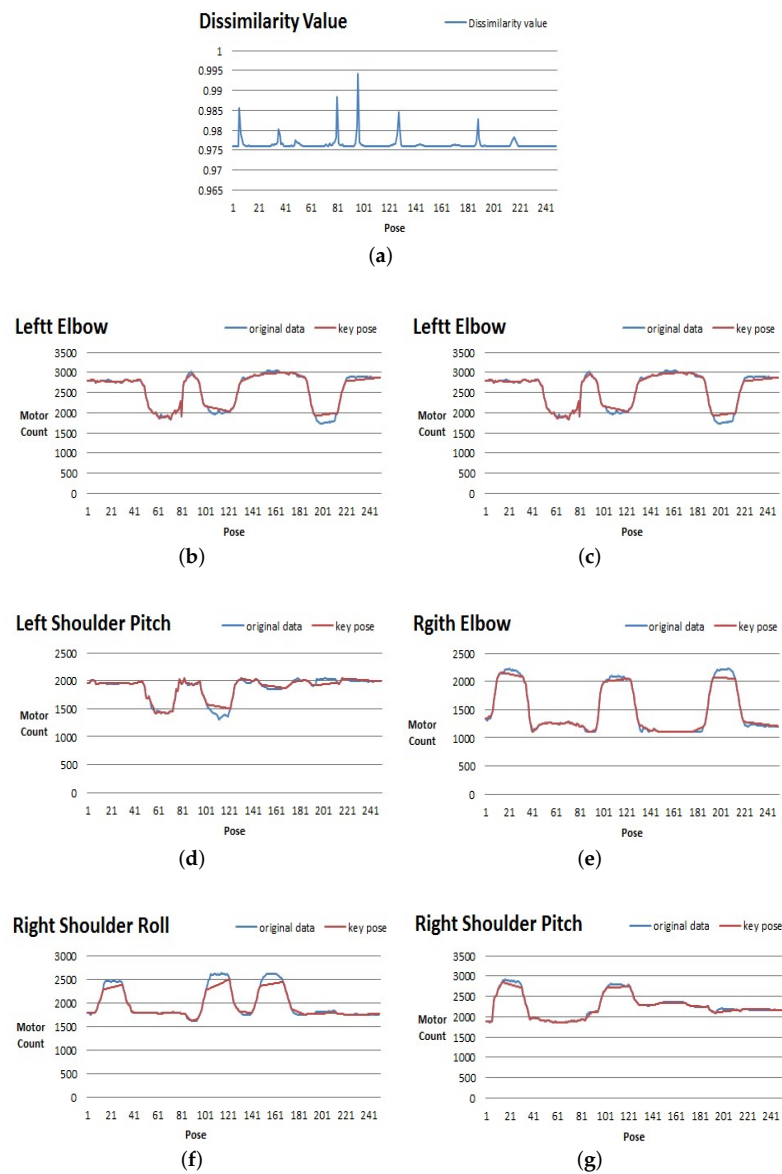


Figure 10. Results of [18] key-pose detection algorithm based on dissimilarity versus original pose data: (a) Calculated dissimilarity value. (b) Motor counts data on Left Elbow. (c) Motor counts data on Left Shoulder Roll. (d) Motor counts data on Left Shoulder Pitch. (e) Motor counts data on Right Elbow. (f) Motor counts data on Right Shoulder Roll. (g) Motor counts data on Right Shoulder Pitch.

The number of key poses and the value of the error can be varied with different values of the threshold, so a suitable value of the threshold, which maximizes the number of key poses recorded and minimizes the value of the error, should be established. Experiments are conducted for that, and the results are shown in Table 4, where different values of threshold, error, and numbers of key poses are listed. To facilitate the work, we specified that any threshold with an error of less than 3 is eligible and we eventually chose 0.35 as our final threshold value.

Table 4. Different threshold and its average of error and amount of key poses.

Threshold	Average of Error	Amount of Key Poses
0.21	0.91062	250 → 100
0.25	1.84542	250 → 89
0.3	2.47993	250 → 78
0.35	2.68958	250 → 69
0.4	3.38649	250 → 58

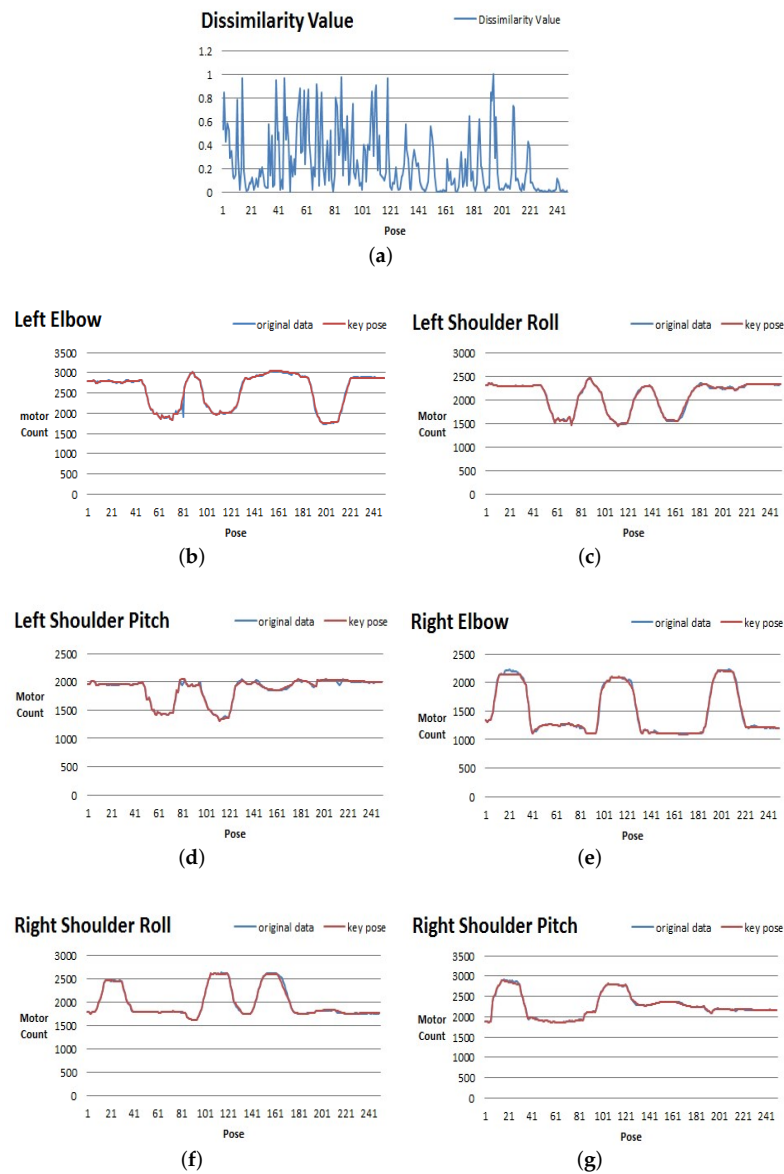


Figure 11. Results of revised key-pose algorithm based on dissimilarity versus original pose data. (a) Calculated dissimilarity value. (b) Motor counts data on Left Elbow. (c) Motor counts data on Left Shoulder Roll. (d) Motor counts data on Left Shoulder Pitch. (e) Motor counts data on Right Elbow. (f) Motor counts data on Right Shoulder Roll. (g) Motor counts data on Right Shoulder Pitch.

3.2. Implementation of Clustering Algorithm

In this section, the implementation of the clustering algorithm on the key poses from the key-pose detection algorithm is discussed and shown. Figure 12 compares the pose data of key poses with and without the clustering algorithm.

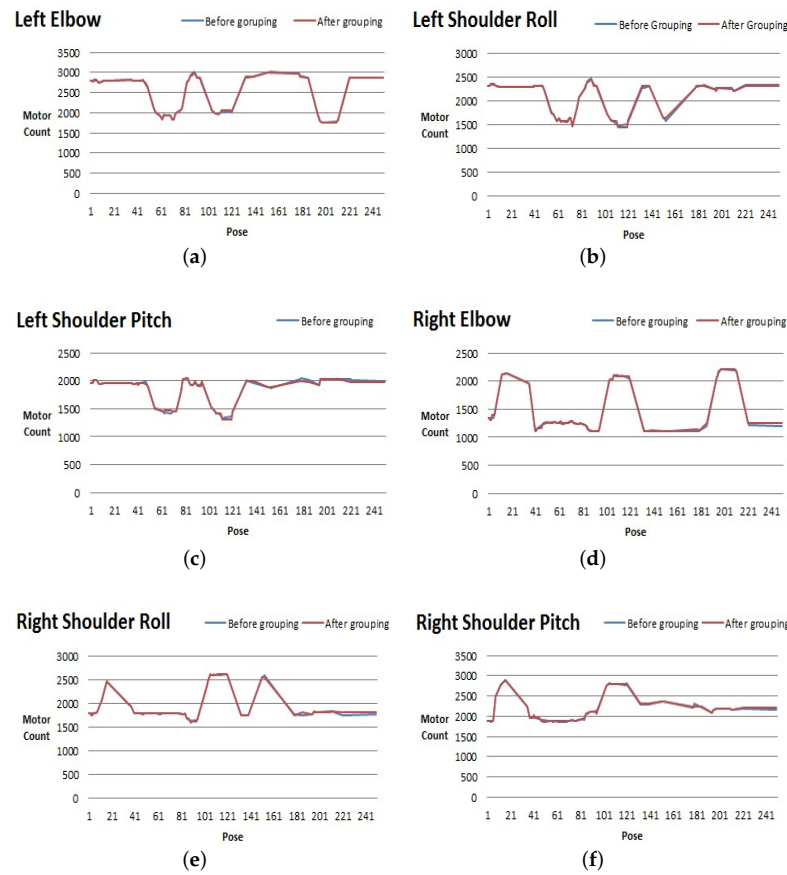


Figure 12. (a) Motor counts data on Left Elbow. (b) Motor counts data on Left Shoulder Roll. (c) Motor counts data on Left Shoulder Pitch. (d) Motor counts data on Right Elbow. (e) Motor counts data on Right Shoulder Roll . (f) Motor counts data on Right Shoulder Pitch.

Table 5 shows the comparison of error values of the key poses processed with and without the clustering algorithm on different parts of the robot. The storage consumption is reduced with the help of the clustering algorithms, while a relatively slight increase in the total error value exists, which is acceptable.

Table 5. Group algorithms comparison table.

Error	on L_E	on L_S_R	on L_S_P	on R_E	on R_S_R	on R_S_P
Without Clustering Algorithm	2.088	2.625	2.159	2.499	4.053	2.711
With Clustering Algorithm	2.263	2.656	2.386	2.695	4.390	3.306

In Table 5, L_E is Left Elbow; L_S_R is Left Shoulder Roll; L_S_P is Left Shoulder Pitch, R_E is Right Elbow , R_S_R is Right Shoulder Roll, R_S_P is Right Shoulder Pitch. All the key poses will be assigned to the suitable group with the group ID after being processed by the clustering algorithm.

3.3. Implementation of Action Emulation, Representation and Memorization

3.3.1. Action Emulation

Figure 13 shows how the humanoid robot replays Action I, which is the result of real-time imitation.

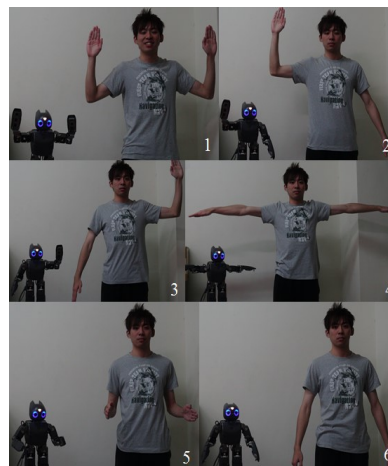


Figure 13. The emulation of humanoid robot.

The motor counts of the motor should be ensured not to damage the motor when implementing synchronous behavior cloning, which means the difference in the structure of different robots must be taken into account. In our experiment, MX-28 is the motor of the humanoid robot, which can rotate in the range of 360 degrees, but it does not ensure that each joint can rotate freely like that, which may cause machinery problems in the robot. Table 6 shows the minimum and maximum scales of the joints on the robot. If such values are not compatible with the number of motor counts, the robot could be damaged.

Table 6. Limited rotation range of the humanoid robot motors.

Name	Minimum	Maximum
L_Elbow	1300	3072
R_Elbow	1024	2700
L_Shoulder_Roll	512	2560
R_Shoulder_Roll	1536	3584
L_Shoulder_Pitch	2048	N/A
R_Shoulder_Pitch	N/A	2048

3.3.2. Action Representation

Figure 14 shows the result of the replay in Action I of the humanoid robot. In the experiment, the action could be entirely duplicated by the robot, and acceptable time consumption was also ensured. In addition, the increase in the error due to the clustering algorithm could be ignored since it is slight enough.

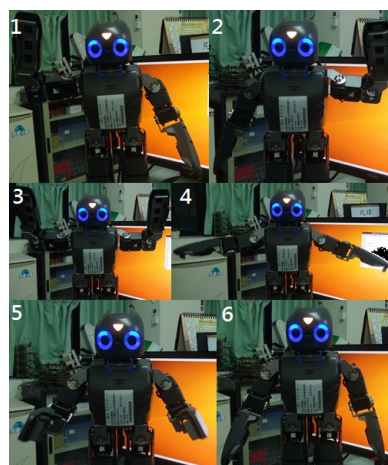


Figure 14. The humanoid robot is replaying action I.

3.3.3. Action Memorization

With the formula calculating the similarity rate of the poses, data processed by the clustering algorithm and the key-pose detection algorithm contain enough information to lead the robot to perform the displayed action, meaning that the function of action memorization is ensured. Table 7 lists the numbers of the original poses and the key poses of Action I to IV. The key poses are the results of the key-pose detection algorithm and the clustering algorithm. As is shown in Table 7, the number of key poses of Action II outnumbers that of Action I by 52, which is great enough to help the algorithm determine that the actions are different. For Actions I to IV, the similarity rates are calculated if the difference of the degree values of joints in key poses falls in a certain range.

Table 7. Amount of the key poses of Actions I, II, III and IV.

Action	Action I	Action II	Action III	Action IV
Number of key poses	250 → 69	250 → 121	250 → 83	250 → 93

After Action I is displayed in front of the RGB-D camera by the user, the data received by the camera will be saved into the database of the robot after being processed by the key-pose detection algorithm and the clustering algorithm. Action II will also be saved into the database because it could be determined to be a new action after being compared to the actions in the database. However, when the Action III is displayed, the data of the action will not be saved because they will be determined as an identical action to Action I. Meanwhile, Action IV will be detected as a new action though it is partially similar to the Action I; for that, the later part of the action is detected as a part of Action IV. In this way, the storage consumption is reduced since no superfluous action will be recorded.

4. Conclusions

In our behavior cloning system, two algorithms are implemented. They are the key-pose detection algorithm and the clustering algorithm. With a fixed limit on the number of key poses that can be stored for different actions, the average error value of the results from the key-pose detection algorithm is nearly 26% lower than [17] and nearly 69% lower than [18]. After assigning each key pose to suitable groups with group IDs, an action can be saved in the form of a series of group IDs, which makes it possible to store nearly 33 times more actions than saving the action in the form of the motor count. Results from the two algorithms ensure acceptable error values while reducing the storage consumption of the action data effectively. In addition, the system implements three basic functions. They are action emulation, action representation, and action memorization. These functions simplify the process of human–computer interaction and expand the user age range of the humanoid robots, which may lend the humanoid robots to become more useful in daily life. Additionally, our system requires less storage space on the robots with the help of the algorithm in the action memorization function.

There still exist problems to be investigated in future work. Currently, the source of the skeleton data is an RGB-D camera in our system, which cannot ensure the accuracy of the processed results, so more advanced multi-view human 3D posture reconstruction methods should be applied in our system to ensure that the original data in our behavior cloning system are not affected by environmental noise or misrecognition. Additionally, if there exist no real-time requirements, then optimization methods, such as genetic algorithms or simulated annealing, can be implemented to find the most suitable threshold parameter in the key-pose detection algorithm with the help of high-performance PCs and robot virtual platforms in order to minimize the error of the recreated action. Furthermore, a module for the equilibrium of the robot should be established to achieve lower body imitation.

Author Contributions: Conceptualization, Q.W., Z.H. and K.-S.H.; Data curation, Q.W. and J.Z.; Formal analysis, Q.W. and H.S.; Funding acquisition, H.S.; Investigation, Q.W. and Z.H.; Methodology, Q.W., Z.H. and H.S.; Project administration, Q.W.; Resources, Q.W.; Supervision, K.-S.H.; Validation, Q.W.; Visualization, Q.W.; Writing—original draft, Q.W., Z.H. and J.Z.; Writing—review and editing, J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by National Natural Science Foundation of China under Grant 92267110, 62076202 and 61976178, Open Research Projects of Zhejiang Lab (NO.2022NB0AB07), Shaanxi Province Key Research and Development Program of China under Grant 2023-YBGY-354 and 2022GY-090, and CAAI-Huawei MindSpore Open Fund (NO.CAAIXSJLJ-2021-041A).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

PC	Personal computer
RGB-D	Red, green, blue plus depth
UDP	User datagram protocol
DOF	Degree of freedom
LCS	Longest common subsequence
$\rho_{i,i+1}$	Calculated dissimilarity value of two poses
$\rho_{i-1,i,j+1}$	Calculated dissimilarity value of three poses
$\beta_{i,j}$	Angle of the joint j in the pose numbered i
$\bar{\beta}_j$	Average value of different angles of the joint numbered j in all poses
σ_{i-1}	Standard deviation of changes in values of joints angles of poses numbered i and $i - 1$
$\Delta\phi_{i-1,i,j}$	Angle change of poses numbered i and $i - 1$ in joint j
$\overline{\Delta\phi}_j$	Average of all the variation in joint numbered j of the poses
T	Time consumed

References

1. Spenko, M.; Buerger, S. *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*; Springer: Berlin/Heidelberg, Germany, 2018.
2. de Andres-Sanchez, J.; Almahameed, A.A.; Arias-Oliva, M.; Pelegrin-Borondo, J. Correlational and Configurational Analysis of Factors Influencing Potential Patients' Attitudes toward Surgical Robots: A Study in the Jordan University Community. *Mathematics* **2022**, *10*, 4319. [[CrossRef](#)]
3. Quevedo, F.; Muñoz, J.; Castano Pena, J.A.; Monje, C.A. 3D Model Identification of a Soft Robotic Neck. *Mathematics* **2021**, *9*, 1652. [[CrossRef](#)]
4. Leonardi, N.; Manca, M.; Paternò, F.; Santoro, C. Trigger-Action Programming for Personalising Humanoid Robot Behaviour. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, UK, 4–9 May 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 1–13. [[CrossRef](#)]
5. Yang, Q.; Steinfeld, A.; Rosé, C.; Zimmerman, J. Re-Examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, Honolulu HI USA, 25–30 April 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1–13. [[CrossRef](#)]
6. Styling Words: A Simple and Natural Way to Increase Variability in Training Data Collection for Gesture Recognition. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Yokohama, Japan, 8–13 May 2021; Association for Computing Machinery: New York, NY, USA, 2021. [[CrossRef](#)]
7. Torabi, F.; Warnell, G.; Stone, P. Behavioral Cloning from Observation. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm Sweden, 13–19 July 2018; pp. 4950–4957.
8. Saponaro, G.; Vicente, P.; Dehban, A.; Jamone, L.; Bernardino, A.; Santos-Victor, J. Learning at the Ends: From Hand to Tool Affordances in Humanoid Robots. In Proceedings of the 2017 Joint IEEE International Conference on Development and Learning and EpiGenetic Robotics (ICDL-EpiRob), Lisbon, Portugal, 18–21 September 2017; pp. 331–337. [[CrossRef](#)]

9. Kosaka, A.; Katakura, T.; Toyama, S.; Ikeda, F. Evaluation of Posture Memory Retentivity using Coached Humanoid Robot. In Proceedings of the HRI '18: Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, Chicago, IL, USA, 5–8 March 2018.
10. Jie, L.; Song, M.; Li, Z.N.; Chen, C. Whole-body humanoid robot imitation with pose similarity evaluation. *Signal Process.* **2015**, *108*, 136–146.
11. Yang, C.; Yuan, K.; Shuai, H.; Taku, K.; Li, Z. Learning Natural Locomotion Behaviors for Humanoid Robots Using Human Bias. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2610–2617. [[CrossRef](#)]
12. Yang, X.; Peng, Y.; Li, W.; Wen, J.Z.; Zhou, D. Vision-Based One-Shot Imitation Learning Supplemented with Target Recognition via Meta Learning. In Proceedings of the 2021 IEEE International Conference on Mechatronics and Automation (ICMA), Takamatsu, Japan, 8–11 August 2021; pp. 1008–1013. [[CrossRef](#)]
13. Wang, C.; Liao, G. Real-Time Pose Imitation by Mid-Size Humanoid Robot With Servo-Cradle-Head RGB-D Vision System. *IEEE Trans. Syst. Man, Cybern. Syst.* **2019**, *49*, 181–191. [[CrossRef](#)]
14. Duranton, C.; Gaunet, F. Behavioral synchronization and affiliation: Dogs exhibit human-like skills. *Learn. Behav.* **2018**, *46*, 364–373. [[CrossRef](#)] [[PubMed](#)]
15. Aljumily, R. *Agglomerative Hierarchical Clustering: An Introduction to Essentials. (1) Proximity Coefficients and Creation of a Vector-Distance Matrix and (2) Construction of the Hierarchical Tree and a Selection of Methods*; Social Science Electronic Publishing: New York, NY, USA, 2017.
16. Wang, Q. A Matching Path Constrained Longest Common Subsequence Length Algorithm. *J. Electron. Inf. Technol.* **2017**, *39*, 2615–2619.
17. Zhao, S.; Wu, Y.; Yang, W.; Li, X. Key Pose Frame Extraction Method of Human Motion Based on 3D Framework and X-Means. *J. Beijing Inst. Technol.* **2017**, *26*, 75–83.
18. Overhill, H. Design as Choreography: Information in Action. *Curator Mus. J.* **2015**, *58*, 5C15. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.