Abstract—A new framework for behavioral learning by multiple humanoid robots using visual motion capture is proposed and analyzed. This novel approach investigates the ability of each robot observer to continuously repeat the actions of the robot immediately preceding it. The importance of this approach is that it removes the need for non-visual interaction between robots. Our experimentation employs two identical humanoid robots that iteratively repeat the motion observed in the other robot, mimicking the results of a long series of robots repeating the actions of an immediately robot. Each basic articulated motion of the humanoid robot is modeled into three layers and registered as learned behavior using color-based landmarks. Three categories of behaviors were examined, using Utgoff’s incremental decision tree algorithm (ITI) to classify the results over time. Close correlation was observed between behaviors of two humanoid robots for middle and high-level descriptions. Detection error occurred in a linear fashion; ITI performed well for behavior classification when the test set of data was acquired temporally close to the sets used to train the tree, even with no error correction efforts. These results demonstrated the usefulness and potential of our unique idea for behavioral repetition and an evolutionary learning scheme.

Index Terms—Reciprocal learning, interaction control in robot systems, behavior modeling, motion analysis.

I. INTRODUCTION

Reciprocal learning is a demanding scheme for human-robot interaction that is facilitated by observing the motions and/or behavior of both a human and a robot. The work in this paper is novel for its attempt towards having one humanoid robot mimic another robot rather than a robot trying to imitate a person. This is possible here because the mechanism of reciprocal learning is bi-directional, meaning that multiple robots can supervise and learn from one another. In order to realize a cooperative and collegial process between humans and robots in the grand scheme of things, automated training for articulated behavior is the key methodology for visual-based interfaces.

Natural interaction between humans and robots has been attempted [1-3], but requires processing capability on the robot’s part [4-6]. When one type of robot has been trained to perform a task, redundant effort on the human’s part would likely be required to train an additional robot to perform the same tasks. Interaction and training require learning that is individualized to the particular make and model of the robot. Thus, direct transfer of code from one type of robot to another may not be possible due to incompatibility of software or differences in the hardware makeup of the robots [7]. However, for types of robots that can be independently programmed by human “teachers”, it may save a great deal of effort to replace the human teacher with a robot teacher. It is not practical for efficiently teaching robots to heavily involve the human in a long-term training phase. During the long training session, it is ideal if humanoid robots incorporate new behaviors via group learning. Theoretically, a single human-robot training session could then be applied towards training an entire fleet of robots whose only compatible interface is a vision-based acquisition system.

We hope to explore the area of continuous repetition in order to compute the learned behavior and then transform it to a different or similar type of humanoid robot using vision-based motion capture. Specifically, this will involve two robots each equipped with a camera and controlling processor (see Figure 1). This will be done using a visual sensing camera attached to the same processor used to control the robot. The robots will then take turns repeating the actions of the other robot. This mimics, in a low-cost manner, the results that would occur if a long series of robots were to continuously repeat the actions of the robot before it. Another application of this method would be an efficient environment where many robots would emulate the actions of a single robot. In the framework proposed here, the human operator does not need to analyze and modify the software for each individual humanoid robot, thus reducing the operator load.
Figure 1. Diagram showing continuous repetition as demonstrated using only 2 humanoid robots.

In the learning phase for training behavior, we would like to implement an on-line learning framework, in which observable instances are incrementally added into the training model. When the degree of behavioral freedom is increased, the feature space tends to explode in a long-term learning system. Our new method prevents this from happening. We use the Incremental Decision Tree (ITI) [8] algorithm to classify the training behaviors. Certain behaviors may follow a common pattern, and we can use entropy information from this kinematical motion to build a common task. We will explain how components of articulated motions can be obtained and discriminated, then applied towards on-line classification.

Visually-based motion capture and inverse kinematics are still required methods to show the proof of concept, not only in the specific scenario of bi-directional robotic supervision in the training phase, but also in the scenario for a robot to mimic a human. In the human-robot interaction scenario, we can efficiently teach the robots instead of programming them, with potentially great time and effort savings. The system proposed here would be capable of learning from humans or from robots (robot-to-robot) with different hardware configurations. First, a robot imitates a behavior of a demonstrator robot, and then it demonstrates the acquired behavior to a different, untrained robot. Through iteration, a common behavior can be transferred to multiple robots with minimal input on the human trainer’s part.

Interaction “error” is a major obstacle to human-robotic interaction. We demonstrate with our series of robot teaching sequences that with adequate and time-incremental training, large variations in the actual behaviors expressed can still lead to correct identification of the originally desired behavior. A caveat of training without applying error correction is that it becomes necessary to utilize incremental algorithms if specific behaviors are to be identified over time. Part of the contribution of this work will involve an evaluation of error propagation across multiple-platform iteration in order to detect motion of each ‘teacher’ robot and to extrapolate the servo motor position data for each individual humanoid robot. Determining the position of articulated objects is very important because the accuracy of such low level motion propagates and influences the higher-level motion accuracy. In our humanoid articulated model, the high level motion represents basic behaviors of a humanoid robot. At the end of the experiment, we will measure the quality of repetition by reading out the position of the final robot’s servo motors and comparing them with the initial controlled robot’s position. These multiple results are used to evaluate the high-level motion transmission across multiple humanoid robots.

The remainder of this paper is organized as follows: Section II describes some prior work done in this field by other researchers. Section III will describe the proposed system architecture, which consists of the humanoid robot, camera, and processor, including the software used. Section IV will describe the method of capturing motion from the observed robot and converting this data into commands that control the servo position. Section V will describe the results obtained from the experiment, and Section VI will conclude the paper.

II. RELATED STUDIES

Although there are few previous studies directly relevant to our proposed framework, the following representative studies show us how to tackle the process of behavioral learning among multiple humanoid robots using motion capture.

Previous work [1, 2, 9] has been done to mimic body structure motion. Some previous research has involved learning human motion to control a robot [4]. Pollard et al. [1] developed a procedure for adapting human motion for the control of a humanoid robot. They studied the capture of upper torso human movement and methods for applying the results to humanoid robot motion. The paper also discussed the limitations of currently available humanoid robot motion. Their main focus was replicating human motion with the greatest accuracy possible within the limitations of the robots’ range of motion. Good imitation of some types of movement were produced, however any motion involving limb overlap or shoulder movement were not well reflected in the humanoid robot’s motion.

Typically, studies on multiple or group robots use mobile-type robots, not humanoid-type robots [10]. Humanoid robots are highly configurable automated devices used in a variety of applications, including entertainment and companionship [3, 11, 12] and use as interactive media agents. Among intelligent agent communities, there is very little research that deals with behavior learning and repetition from humanoid-robot to humanoid-robot. This paper demonstrates how this unique framework can provide promising results, using an evoloutional learning scheme.

Much research expounds the usefulness of human-robot interaction, from companionship [11] to games [13]. It is hard to debate the argument that interaction with robots would be greatly eased if they were able to understand natural human interaction methods [14]. Some believe that an effective
means for teaching robots is to follow a learning process similar to the learning of small children [15] or interaction with caregivers [12].

Natural interaction between humans and robots has been attempted [1-3], but requires processing capability on the robot’s part [4-6]. When one type of robot has been trained to perform a task, redundant effort on the human’s part would likely be required to train an additional robot to perform the same tasks. Interaction and training require learning that is individualized to the particular make and model of the robot. Thus, direct transfer of code from one type of robot to another may not be possible due to incompatibility of software or differences in the hardware makeup of the robots [7]. However, for types of robots that can be independently programmed by human “teachers”, it may save a great deal of effort to replace the human teacher with a robot teacher. Theoretically, a single human-robot training session could then be applied towards training an entire fleet of robots whose only compatible interface is a vision-based acquisition system.

Interaction requires awareness of other robots [7, 16] and implementation of a learning strategy. Past work has been done regarding groups of robots that can work together without a centralized control device [17-20]. Learning strategies must be implemented to implement behavior transference. This paper approaches the topic from the use of primitives [9, 21, 22] to develop a structure for analyzing data from the teacher robot. This data can then be entered into a simulator (i.e. as in [23]). Bentivegna et al. [21] described a system in which the concept of primitives is used in robot learning from observation. Primitives are small parts of any humanoid motion task broken down from a large set of data. In their experiment, primitives are used to break down simple parts of the motions required to play a game of air hockey. An added benefit of primitives is that they can be continuously refined to generate a better result. Amit et al. [9] stated that imitation is an area of great importance because of its ability to simplify a robot's ability to learn new tasks without complex trials. The ability to imitate movement is based on the decomposition of each movement into a set of primitives. This paper suggests the use of a three level hierarchy of information. The bottom level defines base primitives that control simple movements. The next levels define movement specializers and sequence learners that improve the ability to imitate different types of movement through the use of probabilistic models.

Learning techniques can be either directed or supervised (i.e., instructional [24-29] or self-organized [10]. The use of a supervised decision tree method for this paper proved to be most effective because examination of the effects of error drift were desired. The approach here was based on the ID3 and C4.5 variety of decision trees involving reduction of entropy for branch selection [30, 31]. A method for an online updateable tree was needed due to the constant influx of data; ITI [8] was used here, which was developed as an update of ID5R [32].

III. PROPOSED SYSTEM ARCHITECTURE

The overall system integrates the processor, camera, and humanoid robot, and will be based on a compiled C++ executable. This code will give the user a simple interface in which to initially control the robot, and then to activate and deactivate the motion capture portion. The software will be the workhorse of the motion caption, recognition, and replay functions. As shown in Figure 2, the first module is Motion Capture, described in Section III A, and the second module is Redirection, described in Section III B.

Figure 2. Overview of system modules for behavior repetition.

A. Motion Capture

Training articulated behavior using a video camera has been well studied in an off-line manner. The use of vision-based motion analysis is not new; it is rather standard in computational non-contact type measurement systems. For example, some existing 3D motion analysis systems have been already developed in various industrial applications, such as Vicon Motion System [33] A.P.A.S. [34], Motion Analysis [35], and [36-38] in academia. In this paper, we address that autonomous behavior acquisition can be achieved via retargeting basic behaviors. This research also validates that such visual behavior acquisition and classification can be extended from the off-line scheme to an on-line scheme. In this section we will describe how to detect motion in three levels, from low to high, and convert it to servo positions and finally a motion plan.

1) Low Level Motion Detection

One of the most important tasks in motion recognition is identifying and keeping track of a known position on the teaching robot. This is referred to as landmark detection. Since the landmarks are priori-assigned, if the landmarks are
detected in the video sequences, the entire solid articulated body can be fully tracked in the planer space. In our experiment, the landmark detection is accomplished using different colored dots placed in several locations on the robot. The procedure for the low-level motion detection is shown in Figure 3. For each frame of captured video, the image processing software will identify each landmark’s center point. First we run a color filter over the original picture to identify a specific color of a certain sticker to convert to a black and white pixel format. Then to identify the center point we calculate mean distribution of matching pixels in x coordinate.

\[
x_{\text{mean}} = \frac{\sum_{\text{row}=0}^{\text{row}_{\text{max}}} \left( \text{# pixels masked per row} \right)}{\text{row}_{\text{max}}} \tag{1}
\]

\[
y_{\text{mean}} = \frac{\sum_{\text{col}=0}^{\text{col}_{\text{max}}} \left( \text{# pixels masked per col} \right)}{\text{col}_{\text{max}}} \tag{2}
\]

Then recalculate x mean by only averaging data within a given range from the initial mean. Then repeat calculation of mean and mean without outliers in y coordinate. The center will be our reference point and the distance of movement will be calculated by how the center point shifts from image to image.

After identifying the landmark indicators and center points, their location and movement information will be stored in memory along with the time that the video was captured. If the new location information is not significantly different than the previously stored location across the video frames, or if the image is unreadable, the data recording step may be skipped. This will not cause valuable data to be lost because the time is stored, so speed and motion delay data can be reconstructed based on the time stored.

Although only 6 colors of stickers will be used, 12 stickers will be identified by splitting the visual frame vertically such that the 6 stickers on the top of the screen will be considered the 6 arm stickers, and the 6 stickers on the bottom of the screen will be considered the 6 leg stickers.

2) Mid Level Motion Detection

When the landmark detection has completed locating the relevant sticker locations, that data is used to calculate the arm position delta angles. The flowchart for these calculations is shown in Figure 4. First, the A2 and A3 arm segment lengths are determined using the shoulder, elbow and hand sticker locations. Because the camera remains fixed for this experiment, these locations can be taken from any image capture since the A2 and A3 lengths will not change regardless of the current arm location.

> **Figure 3. Low-level motion flowchart.**

> **Figure 4. Flow chart for mid-level motion**

Next, the program will cycle through all captured images detected during the motion capture sequence. For every
image, the shoulder and hand locations will be used along with the previously calculated $A_2$ and $A_3$ lengths to determine $\theta_2$ and $\theta_3$ joint angles. This will take place with every piece of data captured in time, ignoring any bad captures. Refer to later in this section for more information on these calculations.

All of the calculated joint angles will then be compiled together along with the time of the image capture and this will be used to create a general angular motion plan for the robot.

The planning of the trajectory for movement of the right or left arm raise will be done by using the inverse kinematic solution. The general form of the joint matrices is listed below.

$$
\begin{bmatrix}
C\theta_{n+1} & -S\theta_{n+1}C\alpha_{n+1} & S\theta_{n+1}S\alpha_{n+1} & a_{n+1}C\theta_{n+1} \\
S\theta_{n+1} & C\theta_{n+1}C\alpha_{n+1} & C\theta_{n+1}S\alpha_{n+1} & a_{n+1}S\theta_{n+1} \\
0 & S\alpha_{n+1} & C\alpha_{n+1} & d_{n+1} \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

(5)

This will give us individual matrices for each joint, which will allow us to solve for the entire angles for a given position. Once all the joint matrices are calculated, they are substituted into $A_1^{-1} * R_H = A_2A_3$. This is taking the inverse matrix of $A_1$ multiplied by a homogeneous matrix that has two sections, a rotation portion and a position portion.

The $P_x, P_y$ and $P_z$ terms are the position terms, and are what will be using to solve the unknown angles. Where,

$$
C_x = \cos[\theta_1] \\
S_x = \sin[\theta_1] \\
\theta_1 = \text{ArcTan} \left[ \frac{P_y}{P_x} \right] + 180^\circ
$$

(6)

(7)

And for joint 2:

$$
C_2 = \cos[\theta_2] = \text{ArcTan} \left[ \frac{S_1}{C_3} \right]
$$

(16)

(17)

$$
S_2 = \sin[\theta_2]
$$

(18)

And for joint 3:

$$
\theta_3 = \tan^{-1} \left[ \frac{(P_xC_1 + P_yS_1)^2 + P_z^2 - a_2^2 - a_3^2}{2a_2a_3} \right]
$$

(11)

$$
\theta_3 = \tan^{-1} \left[ \frac{S_3}{C_3} \right]
$$

(12)

(19)

(20)

Using the formulas derived above, we will now determine the motion plan necessary to repeat a motion. Let us show an example in the following: the motion capture unit detects a movement of the right finger of the robot from (3,3) to (2.53, 2.73). Also given is that the upper arm length of the robot is 2.25 and the forearm length is 2.75.

First find the servo angles, given that:

$$
\theta_1 = \text{ArcTan} \left[ \frac{P_y}{P_x} \right] = \frac{\pi}{2}
$$

(13)

$$
C_1 = \cos[\theta_1] = 0
$$

(14)

$$
S_1 = \sin[\theta_1] = 1
$$

(15)

And for joint 2:

$$
C_2 = \cos[\theta_2] = \text{ArcTan} \left[ \frac{S_1}{C_3} \right]
$$

(16)

$$
S_2 = \sin[\theta_2]
$$

(18)

And for joint 3:

$$
\theta_3 = \tan^{-1} \left[ \frac{S_3}{C_3} \right]
$$

(19)

$$
C_3 = \frac{(P_xC_1 + P_yS_1)^2 + P_z^2 - a_2^2 - A_3^2}{2A_2A_3}
$$

(20)
Solving for the following positions:

\[ A_x = 2.25, A_y = 2.75, P_x = 3, P_y = 0, P_z = 3, \]

results in

\[ \theta_2 = 1.2074\text{rad}, \theta_1 = -0.2972\text{rad} = -17.0283^\circ \]

Then solving for the final positions:

\[ P_x = 2.53, P_y = 0, P_z = 2.73 \]

This produces

\[ \theta_2 = 1.1901\text{rad}, \theta_1 = -0.4179\text{rad} = -23.9439^\circ \]

Thus, the motion plan requires a move of \( \theta_2 \) by 0.9912 degrees, and \( \theta_3 \) by 6.9156 degrees.

3) **High Level Motion Detection**

Raw behavior identification is the process of taking data directly from landmark detection in tabular form. The data is processed to extract overall motion data acquired in Middle Level Motion.

Movement segment identification is defined as each individual motion, which is called High Level Motion, and assigning it to a body part. For example, movement segment identification would define that the robot moved his left arm up. The full list of movement segments is:

- Left arm up / Left arm down
  - Will be detected by calculating the \( \Delta y \) of the sticker mask coordinates associated with the left arm as detected over a segment of the capture period.
- Right arm up / Right arm down
  - Will be detected by calculating the \( \Delta y \) of the sticker mask coordinates associated with the right arm as detected over a segment of the capture period.
- Left leg left / Left leg right
  - Will be detected by calculating the \( \Delta x \) of the sticker mask coordinates associated with the left leg as detected over a segment of the capture period.
- Right leg left / Right leg right
  - Will be detected by calculating the \( \Delta x \) of the sticker mask coordinates associated with the right leg as detected over a segment of the capture period.

Basic behaviors are defined by combining together High Level motion segments to create a set of predefined movements (i.e., move right arm up, then move right arm down). These High Level motions can then be replayed as preprogrammed movements. Some examples of basic behaviors are:

- Waving: move arms up and then move hands left and right
- Jumping Jack: move hands up and legs out and then back down and in
- Hail a cab: move hand all the way up and stop
- Split: move both legs outward
- Cheer: move both hands upward

In order for robots to learn basic behaviors using video analysis of Middle Level and High Level motion, a well-known decision tree method was used, the Iterative Dichotomizer, version 3 (ID3) [30]. The algorithm for ID3 is shown as Algorithm 1. Another method, Incremental Tree Induction (ITI) [8] is used when necessary to update the

**Algorithm 1. ID3 Decision Tree**

1) Create a single node representing the training samples
2) If the samples are all of the same class, then the node becomes a leaf and is labeled with that class.
3) If attribute list is empty then set the node as a leaf node and labeled with the most common class.
4) Otherwise, select test attribute A, and label the node with attribute A
5) For each value of \( a_i \in A \)
   a) Grow a branch with the test attribute= \( a_i \),
   b) Set \( S_i \) the sample set whose test attribute= \( a_i \),
   c) If \( S_i \) is empty, then attach a leaf node with the most common class
   d) Otherwise, attach the node with a subtree, whose sample set is \( S_i \), and attribute list is the original attribute list minus test attribute.

**Algorithm 2. ITI Incremental Decision Tree**

1) Get new training example
2) Pass training example down branches of existing tree
   a) Update test information at each node
   b) Mark updated node as “stale”
3) When leaf is reached, create a new node if necessary
   a) Implement ID3
   b) Revisit stale nodes recursively and ensure that desired tests are installed
decision tree in an incremental manner. The algorithm for ITI is described in Algorithm 2.

The selection of the test attribute is very important when building the decision tree. Information gain is used [31] to measure the goodness of the selected attribute. The attribute with the highest information gain is chosen as the test attribute for the current node. This approach minimizes the expected number of tests needed to classify an object and guarantees that a simple tree is established.

Suppose our data set is \( S \), which has a quantity \( s \) of sample data. These data belong to \( n \) different classes \( C_i \) \((i = 1, \ldots, n)\). The data set of each class is \( S_i \) \((S_i \subset S)\). Each data has \( m \) distinct attributes. The expected information required to classify the data set is

\[
I(S_1, S_2, \ldots, S_n) = -\sum_{i=1}^{n} p_i \log_2 p_i \quad (22)
\]

where \( p_i \) is the probability that an arbitrary sample belongs to class \( C_i \) and is calculated by \( s_i/s \). Let attribute \( A \) have \( q \) different values \((a_1, a_2, \ldots, a_q)\); therefore, we will have \( q \) branches \((b_1, b_2, \ldots, b_q)\) if attribute \( A \) is used to partition the data set. For each branch \( b_j \) \((j = 1, 2, \ldots, q)\), there are \( s_j \) data belonging to class \( C_j \). The entropy, or expected information based on the partitioning of the data into subsets by \( A \), is given by

\[
E(A) = \sum_{j=1}^{q} \frac{s_j^i}{s} I(S_1^j, S_2^j, \ldots, S_n^j) \quad (23)
\]

where

\[
I(S_1^j, S_2^j, \ldots, S_n^j) = -\sum_{i=1}^{n} p_i^j \log_2 p_i^j \quad (24)
\]

and

\[
p_i^j = \frac{s_i^j}{s_1^j + s_2^j + \ldots + s_n^j}. \quad (25)
\]

A smaller entropy value indicates greater purity in the subset partitions. The expected reduction in entropy caused by attribute \( A \) is defined as:

\[
Gain(A) = I(S_1, S_2, \ldots, S_n) - E(A) \quad (26)
\]

After computing the information gain of each attribute, the attribute with the highest information gain is chosen as the test attribute. The node is then labeled with the test attribute, and the samples are partitioned accordingly.

B. Direction of Identified Behavior

Once a motion has been captured and identified, the software will need to translate the identified motion into a series of servo controls, which will be passed to the robot. These servo controls will be matched in time to the movement of the original robot by a series of angular servo rotations delayed, as necessary, to create the proper speed of motion. The software will use basic serial communications to the second attached robot. This serial communication will make use of basic templated serial communication structures.

1) Robot to Simulation Execution

The first step of the experiment is to detect motion from the first robot and redirect this motion to the Virtual KHR1 simulator. This is done by first running the software that we wrote to detect motion from the robot. The robot must then be controlled by the Heart to Heart software [39] to replay a previously programmed action.

Immediately before the robot initiates a motion, the visual recognition software is started. When the motion is complete, the software will process the recognized motion and will create a macro control file. This file will be loaded into the simulator and played.

The step of processing the data to create a macro control file must be performed as a separate step from actual image
detection because the speed of processing the data would limit the capture rate to an amount below the minimum required to correctly capture the robot’s motions. This limitation can be minimized somewhat by limiting the robot’s motion speeds, but realistic motion must be maintained.

The joint positions measured for this paper are shown in Figure 5, and Figure 6 demonstrates an example how real robot image captured versus simulator screen captures. The steps from robot to simulator are described in Algorithm 3.

2) Dual Robot Execution

Dual Robot Execution requires another Kondo Humanoid Robot to be connected to the system. In this case, the first robot will be directed to perform an action via the same interface with the Heart to Heart software as in the Robot to Simulator execution. This action will be based on a series of simple motions that will then in turn make up the complex action.

The teaching robot will be equipped with colored stickers as was done on the Robot to Simulator execution. The Observer robot will also be equipped with these stickers so that the other robot can capture its motion when the roles are switched. Before the teaching robot is activated to perform an action, the motion capture unit of the second robot will be started and it will capture the motion.

This captured motion will then be analyzed for low-level motion. Once this is captured, the analysis system can collect the low-level motion data and compute the mid- and high-level motion. A motion plan will then be created.

IV. Experimental Results

A. Experimental Setting

The experiments were conducted using a set of interconnected equipment. The main processor of the system was an IBM Thinkpad Model T42 with an Intel Pentium M processor running at 1.5 MHz.

The humanoid robot used in the experiment is a Kondo KHR-1 Humanoid Robot. The interface between the robot and the main processor is via an RS-232 serial port connection. The camera used to detect motion was a Point Grey Firefly camera. The interface between the camera and the main processor is via a Firewire cable. The software portion of the experiment was divided among several parts. First, the image capture portion was code written in C++ using Point Grey API libraries, developed on Microsoft Visual C++.

The Kondo Humanoid Robot was controlled with Heart to Heart robot control software, which allows manual robot control or automated control via saved macro files. The software emulator used to test motion and simulate a second robot for a portion of the experiment was Virtual KHR1.

The experiment was to be performed in a medium light environment with a limited amount of background color or pattern. Although the algorithms could be adjusted to accommodate more background image noise, that is not the goal of this experiment, and thus most background noise will be kept to a minimum.

The robot must be set up with 8 stickers placed in various important locations for tracking. The 4 upper body stickers must be unique colors, as must the 4 lower body stickers. The lower versus upper body stickers will be differentiated based on their relative location. In other words, stickers detected in the top half of the visual detection frame will be associated with the upper body, and vice-versa.

On the upper body, the stickers must be placed at both fingertips, and at the joint between the torso and the arm. The joint between the torso and the arm will be considered the \((P_x, P_y)\) position for all calculations. The fingertip will be considered the \((P_{\alpha}, P_{\beta})\) position. Similarly, on the lower body the stickers will be placed at the tip of the foot and the joint between the torso and the leg.

During the experiment, if there are any time slices in which a sticker location is indeterminable, or dramatically different than the positions in the time slice immediately before or after it, that data will be marked as incorrect, and a more appropriate value will be extrapolated using the neighboring data. This can be done because the speed of the robot is known to not physically exceed a certain amount. Therefore any data that suggests a motion exceeding physical limits must be erroneous and cannot be used.
B. Accuracy of Motion Capture

There are many factors that define the accuracy of the motion capture. We will be calculating accuracy by showing the difference between the teaching robot’s known servo angles and the calculated observed motion, which we call Middle-level motion. This will inherently show the quality of results of the low-level landmark detection as well since this is an integral part of calculating the observed angles. Measurement error must also be calculated in order to estimate error propagations across the behavioral repetitions or iterations.

![Figure 7. An example of a behavior sequence. Frame-by-frame progression of arm positions from Table II are shown for the behavior “Waving”.

These calculation comparisons were only performed on the first iteration of motion. Therefore the expected teaching robot’s servo angles are known. This is an acceptable step to measure the quality of results of the low level motion capture since similar results would be obtained any time given the expected “known” teaching robot position. Iterative loss of quality of results will be analyzed in subsection C., Accuracy of Redirection of Identified Behavior.

Table I describes the data captured for various sampled steps of the robot’s motion, as shown in Figure 7. The table shows the raw landmark detection data for one arm (including shoulder, elbow, and hand positions). The table also shows the calculated joint angles ($\theta_2$ and $\theta_3$). Finally, the table quantifies the results by showing the degrees difference between the known teaching robot’s servo angles and the calculated angles determined from motion capture.

![Figure 8. Actual vs. calculated values for $\Theta_2$](image)

![Figure 9. Actual vs. calculated values for $\Theta_3$](image)

**Table I**

<table>
<thead>
<tr>
<th>#</th>
<th>actual $\Theta_2$ (°)</th>
<th>calc'd $\Theta_2$ (°)</th>
<th>$\Delta$ $\Theta_2$ (°)</th>
<th>actual $\Theta_3$ (°)</th>
<th>calc'd $\Theta_3$ (°)</th>
<th>$\Delta$ $\Theta_3$ (°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>120</td>
<td>120</td>
<td>0.00</td>
<td>160</td>
<td>160</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>130</td>
<td>135.91</td>
<td>5.91</td>
<td>160</td>
<td>159.57</td>
<td>0.43</td>
</tr>
<tr>
<td>3</td>
<td>90</td>
<td>88.67</td>
<td>1.33</td>
<td>160</td>
<td>89.78</td>
<td>0.22</td>
</tr>
<tr>
<td>4</td>
<td>130</td>
<td>136.74</td>
<td>6.74</td>
<td>160</td>
<td>136.28</td>
<td>3.72</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td>37.65</td>
<td>2.35</td>
<td>90</td>
<td>87.57</td>
<td>2.43</td>
</tr>
</tbody>
</table>

Actual indicates the angle programmed for the robot's execution, in degrees. The calc'd angle was computed using the video and the difference between the two is shown under $\Delta$. The # indicates the iteration of the angle measured.
Figure 10. Degree error per iteration for $\theta_2$ across a subset of sample images.

Figure 8 and Figure 9 show actual versus calculated servo angles for $\theta_2$ and $\theta_3$ for the images recorded for Figure 7. As can be seen in both sets of data, the differences seen in the first iteration averaged around 3 degrees. This is a very acceptable number and allows for accurate identification of motion by the learning robot.

C. Analysis for Repetitions

The result of the experiment will be measured by the quality of repetition by reading out the position of the final robot’s servo motors and comparing them with the initial controlled robot’s position. This allows the calculation of a percentage of error in the learned motion versus the teaching motion.

The difference between this section and the results seen in Subsection B is that this section evaluates the results of the motion capture algorithm over a period of many iterative teaching and captures between the two robots. In this experimental setup, the iterations were repeated 5 times to attempt to capture the true effect of degradative quality loss. In the data shown in Figure 10 and Figure 11, only the quality of results for 6 of the captured data points was shown to simplify the display.

It can be seen in Figure 10 and Figure 11 that the error in the detection calculation results is nearly linear. This can be explained by understanding the cause of the visual detection error and simple error propagation. It is known that the calculation error in any given iteration is caused by a combination of imperfect landmark detection as well as calculation resolution loss. Over the course of each iteration, the chance for landmark detection error remains the same as it is not generally affected by location or time. Taking all this into account, it is safe to assume that the linear pattern of data error over many repeated repetitions will continue as long as the landmark detection step does not considerably change.

Although the data quality loss is only linear, it still compounds to a large enough effect to cause serious angle identification problems after only a handful of repetitions. In our testing, once the level of error approached 20 degrees, the high-level motion detection system began to falsely identify motions. From the 1-sided error values listed in Table I, data was assumed to increase in a linear fashion, with potential values varying from a direct linear extrapolation at the same rate as earlier iterations. Values for angle measurements for repetitions past the 6th were simulated by adding the average change in error to the previous value for the repetition, up to a total of 20 simulated repetitions of the initial values. For simplicity, limitations on joint rotations were ignored.

Variance was calculated for separate instances of each repetition by adding a random error of +/- 0% to 100% of the average error (delta as in Table I). The overall trend in angle measurements can be seen in Figure 12, which is, in effect, Figure 10 extended to 20 repetitions using simulated data. It is evident that even with only a linear increase, some error values begin to drift toward very high levels.

Figure 11. Degree error per iteration for $\theta_3$ across a subset of sample images.

Figure 12. Simulated angle error for up to 20 repetitions. Errors for both $\theta_2$ and $\theta_3$ are shown together.
D. Classification of Basic Behaviors

In order to classify behavior sequences, a set of “behaviors” was defined by simulating specific combinations of robot arm angle data. Three behaviors were examined – waving, hailing a cab (hail_cab), and cheering. The sequence of ideal angles for each of these is shown in Table II, and Figure 7 shows the actual sequence of images used to define the behavior “waving.”

Despite the influence of error on the high level motion analysis, the system is still able to classify basic behaviors due to the fact that each behavior sequence is comprised of multiple image capture frames. Utgoff’s ITI decision tree program was used to classify the angle sequences into specific behaviors [8]. The resulting decision tree is shown in Figure 13. The decision tree method was able to classify the behaviors with 100% accuracy when data for all the repetitions is used to train the tree (Figure 13). This is excellent considering the high level of error in later behavior repetitions. However, an issue that arises with the increase in error is that all repetitions must be used to train the decision tree in order to obtain accurate classification. Figure 14 demonstrates the change in the decision threshold that is necessary when output from later repetitions is used for testing. Note that no error correction is applied to the processing between each repetition.

<table>
<thead>
<tr>
<th>Frame #</th>
<th>Waving</th>
<th>Hail_Cab</th>
<th>Cheering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0°</td>
<td>90°</td>
<td>0° 90°</td>
</tr>
<tr>
<td>2</td>
<td>40°</td>
<td>90°</td>
<td>40° 90°</td>
</tr>
<tr>
<td>3</td>
<td>90°</td>
<td>90°</td>
<td>90° 90°</td>
</tr>
<tr>
<td>4</td>
<td>130°</td>
<td>90°</td>
<td>130° 90°</td>
</tr>
<tr>
<td>5</td>
<td>130°</td>
<td>140°</td>
<td>130° 90°</td>
</tr>
<tr>
<td>6</td>
<td>130°</td>
<td>160°</td>
<td>40° 90°</td>
</tr>
<tr>
<td>7</td>
<td>130°</td>
<td>140°</td>
<td>90° 90°</td>
</tr>
<tr>
<td>8</td>
<td>130°</td>
<td>90°</td>
<td>130° 90°</td>
</tr>
<tr>
<td>9</td>
<td>130°</td>
<td>140°</td>
<td>90° 90°</td>
</tr>
<tr>
<td>10</td>
<td>130°</td>
<td>160°</td>
<td>40° 90°</td>
</tr>
<tr>
<td>11</td>
<td>130°</td>
<td>140°</td>
<td>90° 90°</td>
</tr>
<tr>
<td>12</td>
<td>130°</td>
<td>90°</td>
<td>130° 90°</td>
</tr>
<tr>
<td>13</td>
<td>190°</td>
<td>90°</td>
<td>0°   90°</td>
</tr>
<tr>
<td>14</td>
<td>90°</td>
<td>90°</td>
<td>40° 90°</td>
</tr>
<tr>
<td>15</td>
<td>40°</td>
<td>90°</td>
<td>90° 90°</td>
</tr>
</tbody>
</table>

$\Theta_2$ and $\Theta_3$ indicate the sequence of robot arm and shoulder angles for each behavior, shown in order of appearance (Frame #).

Figure 13. Classification results for 20 repetitions of a behavior. Theta2 and Theta3 represent the angles of the upper arm joints, and the _xx suffix indicates the frame number of the behavior sequence.

When the tree for repetitions 0-10 (Figure 15) is used to classify data from iterations 11-20, the accuracy drops from 100% to 45.600%, and is 43.958% when the tree from later repetitions is used to classify earlier data. The tree formed solely from the data for repetitions 11-20 is shown in Figure 16. The gain metric used by the traditional decision tree method chooses a different frame number or cut-off point depending on the amount of error present for the angles.
The issues with later behavior classification could clearly be eliminated if some level of error correction was used in earlier repetitions of behavior. This would be possible if only specific angles or sequences of angles are expected; the robots could then be programmed to compensate for the errors. Since the maximum 2-sided error for the first iteration was less than 15 degrees (2 times \( \theta \) angle number 5), as long as the expected values for joint angles are more than 30 degrees apart, they could be accurately corrected. The accuracy rate for this kind of error correction might theoretically equal 100 for an infinite number of iterations. Since the behavior sequences shown here can still be correctly classified after 20 iterations, it would even be possible to perform error correction at that late point. Error correction would not be appropriate, however, if there are no expected values for the angles or high level behavior. But with linear error, it may be possible to extrapolate the original angles when just two repetitions are recorded and the total number of repetitions is known.

From Figure 15 and Figure 16, the influence of the error on the decision thresholds is visible through the changes in the cut-off points. The decision level for theta 2, frame 5 has nearly doubled to 141.97 from 77.79, for an angle that ideally was 0 degrees. In fact, for theta 2, frame 5, all the ideal angles are less than 142.61 and so theoretically if this tree were used to classify the ideal data, it would misclassify 2/3 of the data. This serves to stress the importance of continually updating a classifier when incremental drift is occurring in the data. An incremental classifier is clearly a useful choice for such a situation.

Figure 17 demonstrates the increase in classification accuracy as the decision tree is incrementally updated (via ITI) with values from each repetition. The test set for this demonstration was comprised of an equal number of simulated data for each repetition (1-20). The increase is generally linear, which is expected given the linear rate of error increase in the data. However, various jumps occur, as between iterations 6-7. This is where the decision tree metric switches the second decision from the 5th frame of theta 2 to using the 6th frame instead (seen in Figure 14).

The jump in accuracy for a different choice of attribute suggests that a metric other than the gain metric of ID3 might be appropriate for data that has a large, consistent, linear drift. A potentially better metric would be one that puts an increased weight on differences between the expected values of each attribute for the labeled output classes. This might improve performance when input data is very noisy or has a linear drift, by choosing attributes that would be the least affected by noise.

Figure 18 shows the distribution of classifications when the test data is processed by each iterative tree in turn (a total of 20 classification attempts). For example, test data sets for repetitions 1-20 are first processed through the decision tree for repetition 1 (labeled Repetition 1 in the figure). Next, this tree is iteratively updated with data for repetition 2, producing a new tree. The test data sets are then processed through the new tree. This continues through the creation and assessment of the tree for the final batch of data. The resulting cumulative classifications are shown in the respective parts of Figure 18; i.e., the behavior Hail Cab was most often classified (correctly) as Hail Cab (Figure 18 (c)), but a portion of the test instances were classified as Waving (Figure 18 (a)), and a smaller set were classified as Cheering (Figure 18 (b)).
Figure 18. Test set classification performance. Each plot shows the ITI classification results for a specific behavior - Waving (a), Cheering (b), or Hail Cab (c). Waving was classified correctly 100% of the time, while the Hail Cab behavior was frequently mis-classified as Waving or Cheering when only the early iterations were used to train the decision tree.

As expected, each behavior is more likely to be classified correctly when a greater number of repetitions are used to train the decision trees. The behaviors most likely to be confused are those that require more than one test for separation. This also seems reasonable because behaviors that are easily separated (using a single test) are likely to have the highest information gain as a decision choice.

V. CONCLUSION

In this paper, we have proposed a new repetition framework for behavioral learning of multiple humanoid robots, using the method of long-term visual-based on-line learning. Rather than requiring massive amounts of computer programming to direct the motions of a humanoid robot, the robot was able to acquire motions using computer vision techniques. This method was extended by allowing additional robots to learn the motions using their own vision systems, thus saving time and effort on the original human’s part.

In the experiment set forth earlier in this paper, the goal was to create a system in which repetition amongst many robots could be tested and refined for the purposes of studying such repetition amongst a long series of robots. This was initially determined to be possible by simulating a long series of robots with only two humanoid robots. The robots would act as parts of two unique systems that would iteratively capture and determine the motion from the previous robot’s motion plan execution. The final goal is transfer the high level or basic behavior across multiple humanoid robots each other using visual motion capture.

Our testing demonstrated the feasibility of this technique. Since high-level motion consists of low-level motion, the error propagations in the low level were evaluated. Given a series of trials, only a small percentage of data quality was lost after each iteration. However, although the results shown by this experiment were good on an iteration-by-iteration basis, the quality of results degraded linearly, and thus would cause a system breakdown after possibly as few as a handful of iterations. The number of iterations possible before the intended motion was not longer preserved could be increased with the use of several refining technique such as a more powerful low level landmark detection sequence, however this only lengthened the time before system breakdown occurred.

The use of basic behavior identification could make the system feasible even for a high number of repetitions. Despite high error rates for extended repetition, decision tree classification techniques can still identify behaviors, as long as the immediately preceding behaviors can be correctly classified. It was necessary, however, to keep the behavior classification tree up-to-date because the value of the angles continued to drift over time.

To fully eliminate the problem of data quality degradation, the system that we have proposed and successfully built would likely need to be enhanced with a method of results feedback and error correction that was not studied in this experiment. Based on the low levels of classification error for basic behavior when all data were used, it may be possible to correct individual angle errors, leading to potentially infinite repetitions with very little angle error.

Possible further studies would also include organizing a flow of repetition, such as interchanging among the humanoid robots, and replaying back to the previous learning behavior. Another application of evolutionary learning schemes (such as the type described in this paper) would be an efficient
environment where more than two robots repeated the action of a single robot. One could imagine a group of “cheerleader” robots emulating the behavior of a “coach” in sequence, or robots produced by different companies being able to “learn” emerging behaviors despite a lack of pre-made programming code for a particular robot model (saving enormous amounts of programming time and increasing cross-platform compatibility of robotic systems).

In conclusion, these studies help to demonstrate the usefulness of evolutionary learning schemes, as well as emphasize points that should be taken into account in order to apply them effectively.

**REFERENCES**


