Robotica (2012) volume 30, pp. 315–332.
doi:10.1017/S0263574711000580

Self-reproduction for articulated behaviors with dual humanoid robots using on-line decision tree classification

Jane Brooks Zurn†, Yuichi Motai†,‡,* and Scott Vento§

†*School of Engineering, Virginia Commonwealth University, Richmond, VA 23284, USA.*
E-mail: brooks@med-associates.com
‡*Med Associates, Inc. P.O. Box 319, St. Albans, VT 05478, USA.*
§*Chelsio Communications Inc. Sunnyvale, CA, USA. E-mail: svento@gmail.com*

(Received in Final Form: May 19, 2011; accepted May 17, 2011. First published online: June 24, 2011)

SUMMARY

We have proposed a new repetition framework for vision-based behavior imitation by a sequence of multiple humanoid robots, introducing an on-line method for delimiting a time-varying context. This novel approach investigates the ability of a robot "student" to observe and imitate a behavior from a "teacher" robot; the student later changes roles to become the "teacher" for a naïve robot. For the many robots that already use video acquisition systems for their real-world tasks, this method eliminates the need for additional communication capabilities and complicated interfaces. This can reduce human intervention requirements and thus enhance the robots' practical usefulness outside the laboratory. Articulated motions are modeled in a three-layer method and registered as learned behaviors using color-based landmarks. Behaviors were identified on-line after each iteration by inducing a decision tree from the visually acquired data. Error accumulated over time, creating a context drift for behavior identification. In addition, identification and transmission of behaviors can occur between robots with differing, dynamically changing configurations. ITI, an on-line decision tree inducer in the C4.5 family, performed well for data that were similar in time and configuration to the training data but the greedily chosen attributes were not optimized for resistance to accumulating error or configuration changes. Our novel algorithm, OLDEX identified context changes on-line, as well as the amount of drift that could be tolerated before compensation was required. OLDEX can thus identify time and configuration contexts for the behavior data. This improved on previous methods, which either separated contexts off-line, or could not separate the slowly time-varying context into distinct regions at all. The results demonstrated the feasibility, usefulness, and potential of our unique idea for behavioral repetition and a propagating learning scheme.

KEYWORDS: Humanoid robots; Multirobot systems; Robot dynamics; Robotic self-replication; On-line pattern recognition.

* Corresponding author. E-mail: ymotai@vcu.edu
Abbreviation: ITI, incremental tree inducer; OLDEX, on-line DElimiter for conteXt

1. Introduction

This is an exciting time for task-oriented humanoid robotics. Science fiction has long provided a vision for applications of humanoid robots, ranging from personal cleaning and service assistants (such as Rosie from the television cartoon *The Jetsons*), to robots who are functionally and visually indistinguishable from humans (the Cylons in the most recent television rendition of *Battlestar Galactica*). Increasingly, complex technology is making it possible for the science fiction vision of robots to be realized.[1] However, the increasing complexity of these robots' tasks also requires increasingly complex training procedures. For example, the dancing humanoid robots at the University of Technology Sydney required approximately 2 h of programming effort for each 5 s of dance, even after developing a platform-specific programming language for the task.[47] Another difficulty is damage or changes to the robot that occur after it has been programmed. Robots are often deployed to locations that are difficult or dangerous for humans to access, such as Mars or underwater. Communication between the robot and humans may be delayed or absent. How does the robot know whether has been damaged? Can it determine this without human input?

The subject of this paper is autonomous identification of time periods where changes occur in the behavior of a robot after the initial training has commenced. A novel robot–robot imitation scenario is employed. Without on-line learning, parameter changes over time could lead to misclassification, unless the context is taken into account. We may not be able to observe changes in the environment surrounding the robot, or prepare for every possible scenario that could affect the robot. We are concerned with two types of changes: (1) changes that occur suddenly (such as loss of a limb), and (2) changes that occur slowly over time (e.g., a failing motor). Our method approaches behavior generation from the perspective of "primitives"—small building blocks of motions that can be combined in various ways to produce more complex behaviors. These building blocks also may serve as a convenient, shared common language between robots. However, if a robot's configuration changes or it becomes damaged, its ability to perform one or more primitives may be compromised. The "shared common knowledge" of the primitive cannot be counted on. Our method makes it possible to update

the definition of a primitive on-line, autonomously. The definition of the primitive can be updated for the robot(s) involved.

To examine these types of changes, we use both a novel imitation scenario between dual humanoid robots, and a new hierarchical decision tree classifier for incremental, autonomous segmentation of time periods between changes. We refer to time periods of consistent behaviors as "contexts." We approach the problem from an observational standpoint—the robot does not need to be aware of its own configuration. It simply observes the behavior of a teacher. When an observable change occurs in functionality, a "context change" has occurred. Using this mutual-imitation scenario, we are able to demonstrate the ability of our algorithm to identify the onset of both sudden and slow changes to the way a robot performs a behavior. Sudden configuration changes are simple to model by removing a limb. Our mutual imitation scenario allows modeling of slow changes by allowing behavior "drift" to occur, unchecked accumulation of error that can eventually disrupt the observed behavior of the robot.

To our knowledge, a physical framework for imitation between humanoid robots has not been previously attempted. Visual learning has been extensively studied in the robot–human imitation domain),[13,19,49,70] but robot–robot imitation has been limited to simulated robots,[17] or nonhumanoid and mobile-type robots.[6,59,61] Also, our approach requires a novel classifier because it incorporates both context identification and on-line learning. Knowledge of context adds a rich dimension to imitation learning—the same measurements observed in one context may have a different meaning in a different context. Our method, OLDEX, can identify when unpredictable, dynamic changes in robot configurations occur, and develops a model that can accommodate the observed changes. It does not need to know what generated the changes (environmental factors, mechanical malfunctions, etc.); the system simply and autonomously observes that changes have occurred and adjusts the classification model accordingly. This makes our method flexible and generalizable across behaviors and environments. None of the above-mentioned studies can incorporate training data from robots whose configurations could change unpredictably during the training period. Also, this research differs from social and multiagent learning because the robots are not technically cooperating with each other on a task (as in Yanco 1994). This system simply creates a model that reflects changes in configurations.

This research is timely because there are several scenarios where external observation of robot behaviors provides benefits that cannot be gained from simple parameter transmission. Programming robots to operate in an open-ended environment is very challenging, and visual observation of performance by robot coworkers could reduce human training loads. This project has recently been undertaken under the EU-funded AMARSi project (Wired, see,[15]). Mutual imitation can also be used to extend current research on robots that are not aware of their current physical configurations due to sparse programming, or robots with a need to continue on-line learning despite random changes to their configuration due to damage.[8,9] Another benefit is that we can learn a great deal about potential malfunction sources using this scheme. A single, small performance error might be almost undetectable, but repeated mutual imitation efforts compound this error to an easily perceptible level. This can accelerate functionality testing by showing which components are sensitive to breakdown. Observation of other robots can also be used as an input by evolutionary games, possibly to study emergence of new behaviors. Also, robot–robot imitation can be extended from the identical robot case to allow imitation between nonidentical agents. Many robots already incorporate visual acquisition systems for their real-world tasks, and few are programmed to communicate directly with robots from other manufacturers. A visual imitation scheme can allow transfer of behaviors by utilizing the robot's built-in visual-to-kinematic conversion abilities.

We developed an on-line machine learning algorithm, OLDEX that can autonomously separate the training period into "contexts"—time periods where the teacher and/or student configurations are consistent. Although there are many definitions of the term "context," it is generally agreed that time variation is an important factor.[16] When the classification of two data instances with similar or identical input parameters depends greatly on the time of acquisition, the system may be described as context-dependent. OLDEX initializes a new context model when it detects a change, which allows the classifier to incorporate the new data into the correct context model. OLDEX can be programmed to draw upon data from other contexts if the teacher has less functionality than the student, yet the new data can still be incorporated to update the model. Thus, with OLDEX and imitation, we gain the ability to autonomously develop models for behaviors despite different and/or changing robot configurations. By selecting the correct model, we may also gain information that the robot can use to evaluate its own physical configuration. To increase OLDEX's accessibility and usefulness to researchers, our implementation is based on the Matlab platform, and the code is available at http://www.gadgetcat.com/.

This paper is organized as follows: section 2 describes some prior work in context research and robot imitation. Section 3 introduces the proposed system architecture, which consists of the humanoid robot, camera, and processor, including the software used. Section 4 will describe the classification of context-dependent behavior, including our proposed method for delimiting contexts on-line. Section 5 provides the experimental setting and results. Section 6 concludes the paper.

2. Related Studies

Since little research has been done regarding context-sensitive robot–robot imitation, this section is divided into four parts: context-sensitive learning, on-line classification, robotic imitation, and interaction between multiple robots.

2.1. *Context-sensitive learning*

The outcomes of context-sensitive systems depend on factors that change over time. Schlimmer and Granger[56] describe one of the earliest works relating to context-dependent learning in their 1986 paper. Their STAGGER

system draws from classical conditioning in psychology, particularly Rescorla's[53] work on contingency to compute two probabilities for each data instance: (1) the likelihood of a feature being associated with a *positive* outcome, and (2) the likelihood of a feature being associated with a *negative* outcome (binary classification). Bayesian formulae are used to compute the probabilities. The system output is a set of Boolean functions and weights. Functional relationships and weights are updated as data are incrementally added. STAGGER and ID3[51] are compared—they are similar in that they construct initially simple, then more complicated representations, and use statistical independence as an evaluation. However, at the time, the STAGGER system was developed, no on-line versions of ID3 had been developed. An incremental extension, ID5R, was introduced in 1989,[63] and the updated method ITI in 1996.[64] The authors also pointed out that on-line incremental capabilities did not necessarily ensure compensation for concept drift. We demonstrate in this paper that ITI's ability to incrementally build the same tree regardless of the input order of the instances makes it capable of accommodating concept drift, but the attributes chosen are not optimized to minimize the impact of data drift. Trees built using the entire dataset are not optimized for entropy reduction of local data regions. Our approach, OLDEX, can be used to explicitly define regions in time (contexts) where the drift in the data is small enough to be optimal for entropy reduction, or regions where a major change has occurred that may require a separate data model.

Widmer and Kubat[67] describe context as the scenario where the generating rules of a dataset are not visible to the system. Their FLORA system can identify similar, recurring contexts, as well as context drift. Window size of a context could be dynamically adjusted, concept definitions stored, and noise robustness was implemented. It forms "description sets" to represent hypotheses for context-dependent concepts and can dynamically focus on new instances data by "forgetting" old data. The method described in this paper improves on FLORA because the on-line induction does not require examination of a window of data, and the importance of old data is automatically decreased simply by adding new data.

Harries *et al.*[23] describe the SLICE system for extracting hidden context in a system. The SLICE system divides a domain into time-delimited concept regions. These regions could be determined in a batch, one-step manner (SLICE-1) by inducing a decision tree classifier (using C4.5) and setting splits on the attribute "time" as locations for context changes. Improved context regions were found by refining the endpoints of the regions using voting methods (SLICE-2). SLICE induces a decision tree classifier by using "time" as an input attribute to the tree. Although this method is effective and produced improved results over STAGGER, it cannot be applied to an incoming data stream in an on-line manner. C4.5 is essentially used as a preinduction method. This is different from how we are applying the decision tree inducer. We wish to induce a decision tree for our actual data, but rather than inject "time" as a variable, we examine the changes in the induced tree over time. Thus, we can find splits in an on-line manner and simultaneously have an updated tree. Also, they found that directly using time as an attribute for the decision tree did not produce an optimal context delimiter. We discuss in Section 4.2 how this may be related to issues of joint probability with C4.5, and how our method OLDEX overcomes this difficulty.

Changes in a classifier over time can be used to indicate context shifts, such as hardware failure or errors. Christensen *et al.* (2008) utilize a neural-network-type framework to identify faults as they occur in a system. OLDEX examines changes in the induced decision tree to identify contexts. Examining on-line changes differentiates our work from the decision tree application of Kim and Lee,[29] who are primarily applying C4.5 as a postprocessing method for error correction, rather than examining the tree to determine what the error means.

Hulten *et al.*[24] present an on-line decision tree classifier (CVFDT) for time-varying contexts. To save computation time, they apply a method of estimating the current error of the inducted decision tree (or subtree) using Hoeffding bounds. When the old tree exceeds the inaccuracy limit, a new tree is induced. Testing the error of the tree is unnecessary in OLDEX because we are constantly inducing new trees in an on-line manner, to monitor changes in the nodes. Once we have the new tree we can use it immediately, and each context has the most accurate tree computed at any time.

2.2. *On-line classification*

Quinlan[52] states that the idea for decision tree induction from training data began in the 1950s[25] and that Friedman's[20] work presents the algorithmic basis for both ID3[50,51] and CART.[11] Two major differences between ID3 and CART are (1) the criterion for choosing decision points, and (2) the method for choosing a tree. ID3 is based on information gain, or entropy, and builds trees in a top-down manner. CART uses the Gini index, which is more commonly seen in financial analysis studies, and chooses from a selection of potential decision trees. ID3 was limited to data instances with nominal-valued attributes (i.e., red, green, blue) and could not separate continuous data (for example, the sequence (2.5, 4.3)). This limitation was addressed in the revision C4.5,[52] which is an extension of ID3 and can process data instances with continuous numerical attributes. ID5R[63] extends these methods to allow on-line incremental induction of trees identical to ID3, for the same data. ITI[64] performs efficient, on-line tree induction for discrete or continuous data attributes and is available on the internet[62]. More detail on the operation of ITI, and how we apply it to our work on context learning, is in Section 4.

Learning techniques can be either directed or supervised[7,36,39,42,57,66] or unsupervised/self-organized.[6,61] The use of a supervised method for this paper proved to be most effective during examination of the effects of error drift and major configuration changes. The approach here was based on the ID3 and C4.5 variety of decision trees, which involve minimizing entropy in order to select tree branches. On-line data incorporation was needed due to the constant influx of data, hence the use of ITI.[64] ITI produces the same tree on-line as off-line. OLDEX maintains this feature, the same models are produced regardless of whether data are incorporated incrementally or all at once.

2.3. Robotic imitation

Among intelligent agent communities, there is very little research that deals with visual-based behavior learning and repetition from humanoid robot to humanoid robot. However, mutual imitation is an early learning stage between parents and infants. Imitation is a preliminary stage in the process of learning to associate behaviors with outcomes; first, a small repertoire of motions is imitated, and then, through repetition and reinforcement, more complex behaviors are formed. Piaget[48] describes typical "baby talk" in the first stage of imitation, where an infant makes a sound, the parent repeats the sound, and then, the child repeats the parent. This is an example of imitation where the role of teacher and student is continuously interchanged. A similar phenomenon is seen for visual tracking, where the parent turns their head and the child repeats the movement. The infant can learn at a very early stage to both imitate and differentiate various motions. Humanoid robots, like infants, are highly adaptable and capable of imitation, and so mutual imitation can be a useful avenue for exploration in robot learning. Amit and Matarić[1] 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.

An important distinction between learning and imitation is that for learning, some judgment must be made regarding whether a behavior is relevant or irrelevant. There must be a specific, known, desired outcome. This research is about the work that comes before that—knowing "something" has changed. We don't know exactly what has changed, because we don't really know the configuration. We don't know whether it affects our goals, because we don't have goals. It really is like imitation in infants—they have no goals, they simply imitate because it is their nature. First, they learn how to imitate, and then, they learn to recognize changes. That is the goal of this paper—to autonomously recognize changes, without requiring judgment or understanding of the particular task involved. Using the newly acquired imitative and change-recognizing skills to accomplish specific goals is the topic of future work.

Humanoid robots are highly configurable automated devices used in a variety of applications, including entertainment and companionship[19,21,33] and use as interactive media agents. Much research expounds the usefulness of human–robot interaction, from aiding in physical therapy[26] to companionship and games.[35] Tani *et al.* (2008) demonstrated a cooperative learning framework between a humanoid robot and a human who provides feedback. They trained a hierarchical neural network model as the human interacted with a robot for error correction. This showed that such observation and interaction could be useful for improving robot learning and that context-dependent behaviors could be described in a hierarchically organized network. This indicates that our approach for classification of behavior using a hierarchical model is appropriate for robot work. Decision trees are well suited for hierarchical descriptions in comparison to neural networks because they are already hierarchically organized, and neural networks can sometimes be structurally indecipherable.

Previous work has been done to mimic body structure motion or human motion for controlling robots (Klingspor 1997).[1,26,49,70] Ijspeert *et al.*[26] demonstrated imitation via control policies for a virtual and a physical humanoid robot, but used sensor data to learn the policies, rather than vision-acquired data. Pollard *et al.*[49] 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 to replicate human motion with the greatest accuracy possible within the limitations of the robots' range of motion. Good imitation of some types of movement was produced; however, any motion involving limb overlap or shoulder movement was not well reflected in the humanoid robot's motion.

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 noncontact-type measurement systems. For example, some existing 3D motion analysis systems have been already developed in various industrial applications by Vicon,[65] A.P.A.S.,[43] Motion Analysis, Inc.,[41] and various academic implementations.[22,58,68] 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 an off-line scheme to an on-line scheme.

2.4. Interaction between multiple robots

Interaction requires awareness of other robots[18,37,59] and implementation of a learning strategy. Past work has been done regarding groups of robots that can work together without a centralized control device.[27,32,36,44] Learning strategies must be implemented to implement behavior transference. This paper approaches the topic from the use of primitives[1,4,5,38] to develop a structure for analyzing data from the teacher robot. This data can then be entered into a simulator, i.e., as in ref. [14]. Matarić[38] as well as Bentivegna *et al.*[4] describe systems 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.

Typically, studies on multiple or group robots use mobile-type robots, not humanoid-type robots.[61] The literature contains a few examples of robot–robot learning,[6,59] but for nonhumanoid (LEGO) robots. Billard's DRAMA architecture (1998) allows a robot student to learn, from a robot teacher, a vocabulary of motions in an arena. But the roles of teacher and student are fixed, whereas in our work the teacher/student roles can be exchanged indefinitely, as in Piaget's baby talk observations. Steels and Vogt[59] implemented an interactive game between LEGO robots in which one robot chose a target object and the second robot attempted to identify it. Although the robots successfully developed schemes to identify objects, recognition was slow and required many repetitions.

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.[46] Some believe that an effective means for teaching robots is to follow a learning process similar to the learning of small children[2] or interaction with caregivers.[33] Natural interaction between humans and robots has been attempted,[19,49,70] but requires processing capability on the robot's part (Klingspor *et al.* 1997).[3,54] 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 a differently configured 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.[18] Also, if a robot's configuration changes after training, it would have to be retrained. 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 toward training an entire fleet of flexibly configured robots whose only compatible interface is a vision-based acquisition system. The work presented in this paper demonstrates preliminary steps toward this possibility.

3. Proposed System Architecture

The system architecture integrates the processor, camera, and a humanoid robot. User interface software was written in C++ and provides a simple interface to control robot movement and motion capture. An overview of the system is provided in Section 3.1. Details of the system operation are organized into modules. The first module, Motion Capture, is described in Section 3.2, and the second module, Redirection, is described in Section 3.3.

3.1. System overview

The experiment in this paper, as mentioned, was a novel paradigm that involved humanoid robot–robot imitation. An elementary machine vision method is used to compute the limb positions of a humanoid "teacher" robot from a video. Then, a second "student" robot imitates the first by implementing the calculated limb positions. This constitutes a single "repetition." The student then becomes the "teacher" for a new robot, and the second repetition begins. This process can be repeated indefinitely, between a series of robots, between only two robots, as shown in Fig. 1, or even with a single robot using a remotely placed video camera. Each repetition may be subject to a small degree of error (which without correction could accumulate indefinitely), or a sudden configuration change could occur (loss of a limb). The experiment in this paper used a single camera and robot; the robot is essentially imitating itself. This removed the influence of physical differences between teacher and student, and limited the source of error to image acquisition, position calculation, and servo motor error.

To test the classifier induced from the observation data, a set of "behaviors" was predefined. Each behavior consists of distinct, ordered combinations of arm and leg positions.

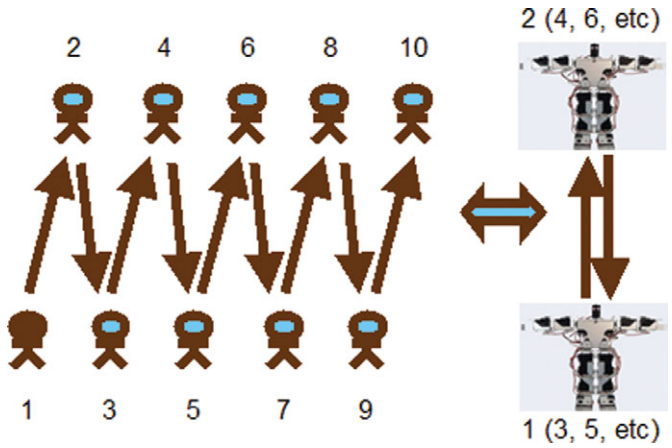

Fig. 1. (Colour online) Diagram showing a continuous sequence of robot imitation from robot to robot (left) or between only two robots (right), much like a robotic game of "Telephone." Error accumulates every time a robot is imitated.

A robot performs a single behavior and the limb and joint positions are recorded, along with the class of the behavior. These data and class are used to induce the classifier. For example, a "Waving" behavior consists of raising the robot arms and moving the hands, while "Cheering" involves moving the entire arm up and down.

To simulate the effects of on-line context changes, a robot's physical configuration was changed by "removing" a part of the body. The hand was removed virtually by deleting the subsequent data for the hand, and continuing the experiment without that part. In this way, the effects of physical context changes could be observed.

The ability to imitate movement is based on the decomposition of each behavior into a set of primitives. This paper suggests the use of a three-level hierarchy of information. In the bottom level, base primitives are defined that that control simple, low-level movements. We defined six simple primitives based on arm positions. In the middle level, we use kinematic equations to solve for joint angles to execute the primitives. In the highest level of the hierarchy, behaviors are defined using time-ordered sequences of primitives.

Due to various on-line system changes (such as error accumulation or limb loss/malfunction), the limiting values for each primitive may evolve over time. We showed that when the classifier is induced from data in only one context, the model may generalize poorly for other contexts. The off-line classification methods used for ref. [17], or Calinon and Billard (2005) produced static models, which required the entire dataset for retraining. This could cause difficulties for sparse but increasing training data or dynamically changing configurations. The probabilistic classification method used in this paper (ITI incremental decision tree) could incorporate new data on-line, and so compensate for physical configuration changes if and when they occurred. Our novel OLDEX algorithm was able to identify the time point for these context changes. A sudden, isolated context change could be cause by malfunction of a part. A consistent frequency of context changes might indicate a need for detrending. Thus, the context changes

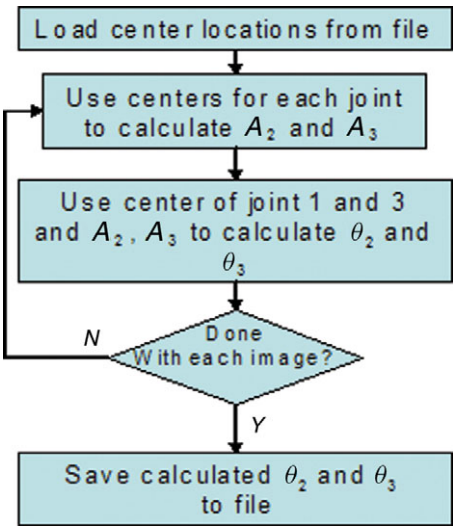

Fig. 2. (Colour online) Flow chart for mid-level motion.

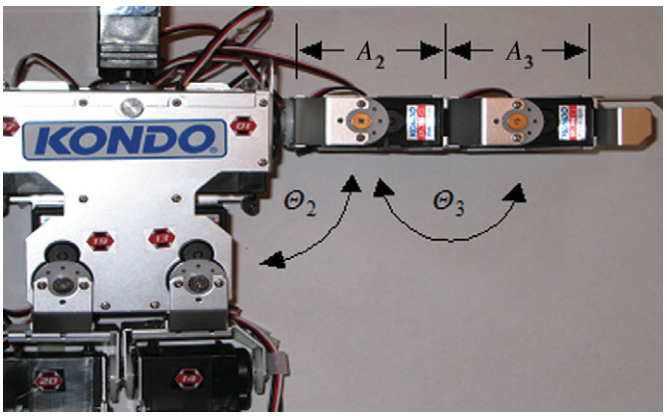

Fig. 3. (Colour online) Robot arm segment lengths A_2, A_3 and joint angles θ_2, θ_3. The lengths and angles are used for high-level behavior classification.

can be used to perform autonomous system self-monitoring without external reinforcement.

3.2. Motion capture

The motion capture process described here is divided into three parts: a low-level process, mid-level process, and a high-level process. This mirrors, in a broad way, Piaget's[48] progression of imitation, starting with simple tasks, and proceeding to more complex combinations. The steps used in this paper operate as follows.

3.2.1. Low-level motion detection. To detect low-level motion, landmarks are assigned *a priori* using different colored dots placed in several locations on the teaching robot. Twelve stickers mark the limbs and joints, but only six unique colors were needed—six for the top half of the screen ("arm" stickers) and six for the lower half ("leg" stickers). To detect each landmark's center point, we first run a color filter over the original picture to isolate a specific sticker color. After masking the isolated color, we calculate the mean x- and y-coordinates from the pixel locations in the mask.

To reduce the effect of outliers, the x- and y- positions are then recalculated by averaging only the masked pixels within $+/-$ 20 pixels of the calculated mean.

The movement distance between frames is calculated using the center points, and both the movement and center points are stored in a time-stamped file. To save time and memory, recording is skipped when images are unreadable or motion between captures is insignificant.

3.2.2. Mid-level motion detection. In mid-level motion, an operational space approach is applied.[28] The center locations found in low-level motion are used to solve for the arm lengths and joint angles. The flowchart for computing the lengths and angle changes is shown in Fig. 2. First, the A_2 and A_3 arm segment lengths (see Fig. 3) are determined from the shoulder, elbow, and hand sticker midpoints. The camera and robot distance remains fixed in the z-axis. Using these positions, the joint angles θ_2 and θ_3 are calculated. To simplify computation, "bad captures" or missing data are ignored. This time-stamped angle data will be used to create a general angular motion plan for the robot. The planning of the trajectory for moving the right or left arm will be computed using the inverse kinematics solution. The general 3D form of the joint matrices is shown in Eq. (5)

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

where C and S represent Sin and Cos, θ is the angle with respect to the XY-plane, and α is the angle with respect to the Z-plane. The $n+1$ subscripts indicate that the angles corresponding to the new positions. This will give us individual matrices for each joint, which will allow us to solve for the entire set of angles for a give position. Once all the joint matrices are calculated, they are substituted into $A_1^{-1} *^R T_H = A_2 A_3$. This involves taking the inverse matrix of A_1 multiplied by a homogeneous matrix that has two sections: a rotation portion and a position portion (see Eq. (6))

$$\begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^R T_H. \quad (2)$$

The P_x, P_y, and P_z terms are the position terms and will be used to solve for the unknown angles (see Eqs. (7)–12))

$$C_x = \mathrm{Cos}[\theta_x], \quad S_x = \mathrm{Sin}[\theta_x], \quad \theta_1 = \mathrm{ArcTan}\left[\frac{P_y}{P_x}\right], \quad \theta_1 = \theta_1 + 180^\circ, \quad (3)$$

$$S_2 = \frac{(S_3 a_3 + a_2)(P_z) - S_3 a_3 (P_x C_1 + P_y S_1)}{(C_3 a_3 + a_2)^2 + S_3^2 a_3^2)}, \quad (4)$$

$$C_2 = \frac{(C_3a_3 + a_2)(P_xC_1 + P_yS_1) + S_3a_3P_z}{(C_3a_3 + a_2)^2 + S_3a_3^2)}, \quad (5)$$

$$\theta_2 = \tan^{-1}\frac{(C_3a_3 + a_2)(P_z) + S_3a_3(P_zC_1 + P_yS_1)}{(C_3a_3 + a_2)(P_xC_1 + P_yS_1) + S_3a_3P_z)}, \quad (6)$$

$$C_3 = \frac{(P_xC_1 + P_yS_1)^2 + P_z^2 - a_2^2 - a_3^2}{2a_2a_3}, \quad (7)$$

$$\theta_3 = \tan^{-1}\frac{S_3}{C_3}. \quad (8)$$

3.2.3. High-level motion detection. For high-level motion, motion segments are defined as specific combinations of robot limb positions. The specific motion segments are defined as left arm up/down, right arm up/down. They are determined by calculating the corresponding midpoints with respect to each frame of the time period. Ordered combinations of these motion segments constitute behaviors.

3.3. Redirection 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 (the "Teacher") 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 (the "Student"). This serial communication will make use of basic templates for serial communication structures. Two methods of redirection are described. The first is a Robot-to-Simulation method, where the "Teacher" robot is a Kondo Humanoid robot, and the "Student" is a virtual robot existing in the Virtual KHR1 simulator. The second is a Dual Robot method, where both the Teacher and Student are physical robots.

3.3.1. 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[31] 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. Figure 4 demonstrates an example of real robot images captured versus simulator screen captures. The steps from robot to simulator are described in Algorithm 1.

Algorithm 1. Robot-to-Simulator Procedure

1) Acquire video of robot performing specific movement
   a) Divide video sequence into series of frames
2) Calculate the sticker center position from the video frames
   a) Use the mean calculation of sticker center pixels and eliminate outliers
3) Calculate angles based on sticker position
   a) Feed sticker position back into movement equation for robot
   b) Record angles
4) Feed angles back into simulator to check movement
   a) Movement should mimic the robot's original movement

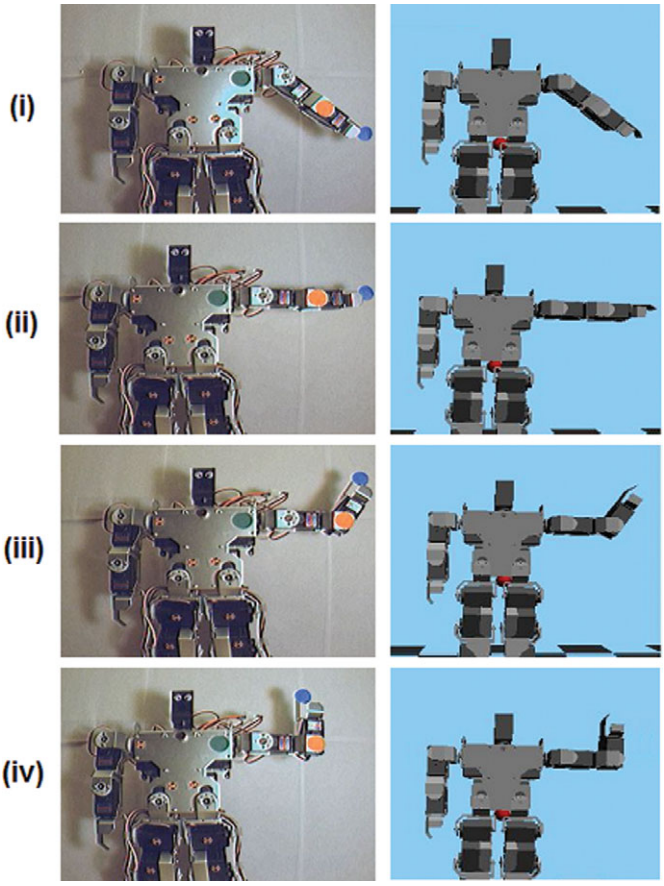

Fig. 4. (Colour online) Real robot image captures vs. simulator screen captures—both using the same real video stream data.

3.3.2. Multiple robot execution. Multiple robot execution requires additional Kondo Humanoid Robot(s) to be connected to the system. However, for the experiments implemented in this paper, only a single physical robot was used. To simulate an experiment with multiple robots, the acquisition steps of Section 3.1 were first performed for the "Teacher" robot. The camera/computer system acquired and processed the data for the Teacher, using the Heart to Heart software and interface. Then, the *same* robot

switched to the role of "Student." The computed motions were passed to the robot from the camera/computer system. This limited the source of error to position calculation and implementation, rather than interrobot differences in joints and sticker placement. It also made it possible to emulate the results of a procession of robots imitating each other, without requiring multiple robots.

3.3.3. On-line recognition and compensation for context changes. The simplest scenario for robot–robot imitation involves two identical robots. However, this could be considered a trivial scenario since one might simply transmit the machine control code from one to the other. A nontrivial scenario, however, is robot–robot imitation between nonidentical robots, particularly if the configurations change after the training process has begun. We describe here an on-line strategy for behavior imitation that compensates for physical changes that may occur to one or more robot structures *after* the start of the exercise.

If a teacher robot loses part of a limb after the beginning of an imitation session, it is considered a context change. We must address how this context change affects the imitation process. How does a robot imitate a high-level behavior produced by a nonidentical robot? We use an on-line learning method (OLDEX) that identifies when context changes occur (such as limb loss or malfunction). Once a context change occurs, and the robots are no longer identical, how do we objectively define a behavior? Two approaches are (1) perform an approximation of the current motion, within the constraints of the anatomical differences, and (2) use a machine learning method to identify the most important aspects of the behavior, and implement those in the nonidentical robot. The difference between these approaches is that (1) simply replicates previous motion, saving computational effort at the potential expense of accuracy, and (2) performs a more intelligent analysis of what makes that motion "important." By understanding what makes the motion important, we have a better idea of what needs to be replicated. Our strategy involves both methods, by (1) first processing the image, then (2) inducing a decision tree from the analyzed image.

Initially, a model for the most important features of the behavior is induced using a machine learning method (here, ITI). When a context change is identified, the previous model is saved and associated with the robot configuration that last contributed to the model.

If the number of parts has changed, first we check to see if this configuration has previously been observed. If it has, then a model will exist in storage. We incrementally update the appropriate model using the new data. If the configuration has not been seen before, a new model is created. The most recently used model is chosen as an initial model, and modified with respect to the new number of attributes. We are using a tree-structured model—each part corresponds to a training attribute. The new model is created using either (a) or (b):

(a) If a part has been removed, the corresponding attribute is deleted and tree nodes based on that attribute are deleted.
(b) If a part has been added, the number of attributes in the training model is augmented on-line.

The new data is then incrementally incorporated into the new model.

This method of creating/updating different models for different configurations has several benefits. Classification is more accurate when a configuration-specific model is used (see Table II in the results section). There is also benefit from basing the new model on the previous model. Since the new configuration developed from the previous configuration, there will be at least some similarity between the two (even if multiple or cascading part failures occur). By basing the new model on the previous model, but updating it separately, the algorithm can selectively modify only the aspects that have changed.

4. Context-Dependent Behavior Classification

This section discusses the classification method for robot behaviors, performed by an independent observer. Because of the compounding effects of error, the classification parameters are affected by time, and thus, the system becomes a model for a time-varying context. We will arbitrarily describe two time contexts, "early" observations and "late" observations. Section 4.1 discusses identification and classification of behaviors, and Section 4.2 describes our contribution of on-line decision-tree-based context separation.

4.1. Behavior classification

Behaviors are defined here as a 15-frame sequence of high-level motions. Groups of 15 frames were sent as a unit to the classifier. If the user requires longer sequences for behaviors, or sequences of unequal length, the length of the sequences can be set to the longest length and extra frame data simply set to "missing." This sequence length ensured that the combination and ordering of the primitives for high-level motions would be unique for each behavior. The start and stop times of the sequences were provided externally. 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. Examples of basic behaviors in this paper are following:

(a) Waving: move arm up and then move hand left and right.
(b) Hail a cab: move hand all the way up and stop.
(c) Cheer: move hand upward.
(d) Exercise: move arm all the way up and down, repeat.
(e) Flapping: move arm up and down in limited range.
(f) Weights: move arm in short range.
(g) Curl, early, mid, and late: Same as Weights, but for one frame (early, mid, or late in sequence) the hand is bent inward at a tighter angle.

The Curl behaviors are used to demonstrate system performance between very similar sequences. They are nearly identical, with only one differing frame. This makes

Algorithm 2. ID3-type Decision Tree Inducer

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 3. 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-type Decision Tree Inducer**
   b) Revisit stale nodes recursively and ensure that desired tests are installed

it a more "difficult" behavior to distinguish in the presence of drift and error.

In order for robots to learn basic behaviors using video analysis of middle-level and high-level motions, ITI[64] was used. ITI produces the same decision trees regardless of whether the dataset is incorporated incrementally or in batch mode (all data at once). It evolved from the ID3[51] and ID5R[63] decision tree inducers and can handle both nominal and discrete data. ID3 is described in Algorithm 2, and Algorithm 3 shows how ITI extends the method for incremental use.

The selection of the test attribute is very important when building the decision tree, and so we will delve into a derivation of information gain, which is used by ID3-based classifiers to measure the goodness of the selected attribute.[52] 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. Schlimmer and Fischer[55] suggested use of the *K–S* distance measure as an incremental equivalent to information gain. It has been shown to produce equivalent results to entropy-based tests.[64]

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

$$I(S_1, S_2, \ldots, S_n) = -\sum_{i=1}^{n} p_i \log_2 p_i, \tag{9}$$

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 dataset. For each branch $b_j, (j = 1, 2, \ldots, q)$, there are s_{ij} data belonging to class C_i. The entropy, or expected information based on the partitioning of the data into subsets by A, is given by

$$E(A) = \sum_{j=1}^{q} \frac{s_1^j + s_2^j + \cdots + s_n^j}{s} I\left(S_1^j, S_2^j, \ldots, S_n^j\right), \tag{10}$$

where

$$I\left(S_1^j, S_2^j, \ldots, S_n^j\right) = -\sum_{i=1}^{n} p_i^j \log_2 p_i^j \tag{11}$$

and

$$p_i^j = \frac{s_i^j}{s_1^j + s_2^j + \cdots + s_n^j}. \tag{12}$$

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

$$\text{Gain}(A) = I(S_1, S_2, \cdots S_n) - E(A). \tag{13}$$

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.

It is important to note that for an attribute with continuous numerical values, the number of potential branches is limitless. Hence, ID3 only deals with discrete-valued attributes. C4.5[52] approaches this issue through the observation that the number of *actual* values in a training set for attribute A are finite, thus $(a_1, a_2, \ldots, a_q)$ are limited to values seen in the test set. A binary search is executed to find a suitable cutpoint for performing a threshold test. This simple approach makes it possible to apply the ID3 algorithm to continuous-valued attributes.

4.2. OLDEX: On-line DElimiter for conteXt

In order to identify times for context breaks in an on-line manner, we examine the decision trees as they are updated with new iterations of data. Our hypothesis is that identifying a new context at a time where the attribute choice changes for a decision node will improve classifier accuracy in the presence of context drift. This provides an automatic method for partitioning a data stream. Harries *et al.*'s[23] SLICE method approached off-line context division in a similar manner, by building a decision tree using the time of a data instance as an attribute. Points in time that were

chosen as test attributes have maximum gain (Eq. 17), and so partitioning the set at those points should improve test set accuracy. However, the points found did not always produce the most optimal locations for context breaks. They were able to improve the method by creating a search window around those times. This likely occurs because the summation of probabilities computed in the entropy calculation assumes that the distributions are independent. Because we are dealing with context (time) dependent data, we know for a fact that independence does not exist (Eq. 18)

$$(p_i^j|\text{time} = t) \neq (p_i^j|\text{time} = t+1), \tag{14}$$

assuming that the total number n of classes i remains constant over time. Therefore, testing with respect to the attribute time will produce a biased result.

Our method for determining the location for a context split involves finding a time where the attribute choice changes for the tree. The information at time $= t$ is

$$I\left(S_1^j, S_2^j, \ldots, S_n^j\right) = -\sum_{i=1}^{n}\left(p_i^j|\text{time} = t\right)\log_2\left(p_i^j|\text{time} = t\right)$$
$$= \sum_{i=1}^{n}\log_2\left(\left(\frac{1}{p_i^j|\text{time} = t}\right)^{(p_i^j|\text{time}=t)}\right) \tag{15}$$

and

$$E(A) = \sum_{i=1}^{n}\sum_{j=1}^{q}\frac{s_1^j + s_2^j + \cdots + s_n^j}{s}$$
$$\times \log_2\left(\left(\frac{1}{p_i^j|\text{time} = t}\right)^{(p_i^j|\text{time}=t)}\right). \tag{16}$$

Rather than testing only the influence of the attribute *time*, by looking for an on-line change in the attribute selected by the decision tree, we are examining the cumulative effects of time across all of the attributes. The following derivation shows the change in entropy $\Delta E(A)$ between time t and time $t+1$:

$$\Delta E(A) = \sum_{i=1}^{n}\sum_{j=1}^{q}\frac{s_1^j + s_2^j + \cdots + s_n^j}{s}$$
$$\times \log_2\left(\left(\frac{1}{p_i^j|\text{time} = t+1}\right)^{(p_i^j|\text{time}=t+1)}\right)$$
$$-\sum_{i=1}^{n}\sum_{j=1}^{q}\frac{s_1^j + s_2^j + \cdots + s_n^j}{s}$$
$$\times \log_2\left(\left(\frac{1}{p_i^j|\text{time} = t}\right)^{(p_i^j|\text{time}=t)}\right)$$
$$= \sum_{i=1}^{n}\sum_{j=1}^{q}\frac{s_1^j + s_2^j + \cdots + s_n^j}{s}$$
$$\times \log_2\left(\frac{\left(p_i^j|\text{time} = t\right)^{(p_i^j|\text{time}=t)}}{\left(p_i^j|\text{time} = t+1\right)^{(p_i^j|\text{time}=t+1)}}\right). \tag{17}$$

Algorithm 4. OLDEX: On-Line DElimiter for conteXt

1) Get training examples for current iteration
2) Implement **ITI**
3) Check for changes in ITI tree
   a) If data is missing from an attribute (after attempted compensation), mark a context delimiter.
   b) Compare current node tests to those from "old tree"
      i) If "old tree" doesn't exist, save current tree as "old tree"
      ii) Otherwise, if tests use different attributes, mark a context delimiter and create a new model
4) Delimit Context?
   a) If no context delimiter,
      i) Save "current tree" as "old tree"
   b) Otherwise, if context is delimited here,
      i) Search sets of previous models
      ii) If no match among previous models, create a new model. Otherwise, set current model to matching model.
      iii) Implement ITI for current training examples
      iv) Save "old tree" as final tree for previous context
      Save tree from current iteration data as "old tree"

Therefore, the change in entropy between time t and time $t+1$ is proportional to the log of the probability ratios, and the conditional probabilities are taken into account.

The tree produced by the decision tree inducer shows the direct result of the entropy calculation for each iteration. By examining the tree nodes for changes over time, we have a relatively simple means of identifying large changes in $\Delta E(A)$ with respect to time. This can be used as a breakpoint in time for context changes. We call this method OLDEX; the algorithm is shown in Algorithm 4. We show in Section 5 that this strategy is more effective for finding the delimiting times in this slowly varying context than using time (off-line) as an attribute in the decision tree inducer and gives similar results as when a window in time is examined, with the ability to identify the delimiting time on-line.

Algorithm 4 describes the process for delimiting the context on-line. After data from the current iteration is acquired, the tree is updated and called the "current tree." Two conditions can lead to a context change flag. The first (simpler) condition is the sudden absence of data for an attribute. If data cannot be acquired for a specific attribute for a certain amount of time, the system defines a "context change" (missing attribute) and defines a new model. The other condition for a context change is defined as follows. The attribute choices for the "current tree" are compared to those for a tree induced from the previous iteration data

(if this is not the first run). If the choice of attributes has changed between the current tree and the last induced tree, the change in entropy with respect to time has been high enough to affect the information calculations, and we mark the current iteration as a context delimiting point. The tree from the previous context is saved, and data from the current iteration are used to train a new "current tree" for the new context. After the context tests have been performed, the current tree is saved as the "old tree" for purposes of the next comparison.

5. Experimental Results

Here, we describe the experimental process of implementing and testing the robotic imitation, and present the context identification results. Section 5.1 specifies the experimental setting, including hardware parameters. Section 5.2 discusses the contributing factors for the accumulating error scenario. In Section 5.3, the accumulating error is analyzed. Section 5.4 shows the behavior classification results in the context of the accumulating error. Section 5.5 demonstrates the positive contribution of our OLDEX toward improving behavior classification accuracy and shows that it has superior results in comparison to a previously published method, SLICE.[23]

5.1. 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, which was interfaced to the main processor via an RS-232 serial port connection. The camera used to detect motion was a Point Grey Firefly camera, connected via a Firewire cable.

The software portion of the experiment was divided among several parts. 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 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 would be the focus of a different experiment; thus, most background noise was kept to a minimum.

On the upper body, tracking stickers were 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_{x0}, P_{y0}) position for all calculations. The fingertip was considered the (P_x, P_y) position. Similarly, on the lower body, tracking stickers were placed at the tip of the foot and the joint between the torso and the leg.

During the original experiment, if there are any time slices in which a location was indeterminable, or dramatically different than the positions in the time slice immediately before or after it, that data point was marked as incorrect and a more appropriate value was interpolated using the neighboring data. This was done because the speed of the robot was known to not physically exceed a certain amount. Therefore, data that suggests a motion exceeding physical limits must be erroneous and cannot be used. For the later configuration change experiments, measurements were simulated using these same data, but simply deleting values for "missing" limbs.

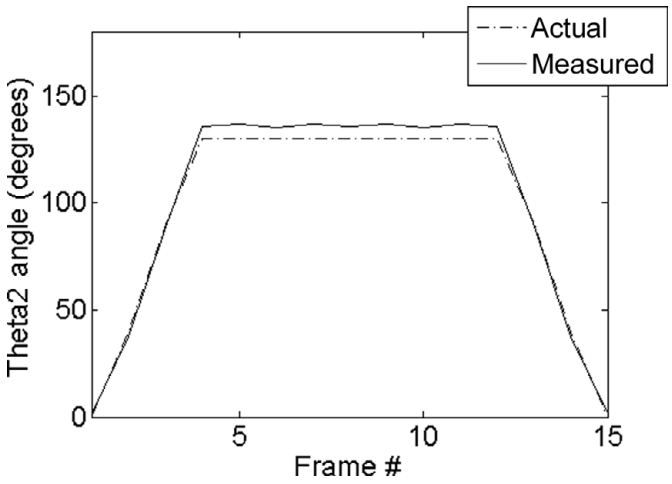

Fig. 5. Actual vs. calculated values for θ_2.

5.2. Accuracy of motion capture

There are many factors that affect the accuracy of the motion capture process. We define accuracy in this paper as the difference between the teaching robot's known servo angles and the calculated observed (Mid-Level) motion. This will inherently be affected by the low-level landmark detection as well since it 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.

Table I describes the data captured for various sampled steps of a sample of robot motion. The table shows a set of programmed angles (θ_2 and θ_3) for a teacher robot, and the observed angles, as calculated by the system. The angular differences in these results are quantified under Δ. Figures 5 and 6 show a plot of actual versus calculated values for θ_2 and θ_3 in the recorded images. As can be seen in both sets of data, the differences seen in the first iteration averaged around 3°. This amount of error does not present a problem for classification in early repetitions. However, the error accumulates as the number of repetitions increases, which, if uncorrected, will eventually affect classification. The result of the experiment will be measured by the quality of repetition by reading out the final positions of the robot's servo motors and comparing them with the initial, controlled servo angles. This allows the calculation of a percentage of error in the imitated motion versus the teaching motion.

5.3. Analysis of error accumulation between repetitions

In this section, we examine the accumulation of error that occurs over a period of iterative teaching and imitation sessions. In this experimental setup, the iterations were

Table I. Selected image calculation errors. 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 Δ. The # indicates the iteration of the angle measured.

Angle #	Actual $\theta_2(^\circ)$	Calc'd $\theta_2(^\circ)$	Error $\Delta(^\circ)$	Actual $\theta_3(^\circ)$	Calc'd $\theta_3(^\circ)$	Error $\Delta(^\circ)$
1	0	1.25	1.25	90	87.36	2.64
2	130	135.91	5.91	90	86.7	3.3
3	130	135	5	160	159.57	0.43
4	90	88.67	1.33	90	89.78	0.22
5	130	136.74	6.74	140	136.28	3.72
6	40	37.65	2.35	90	87.57	2.43

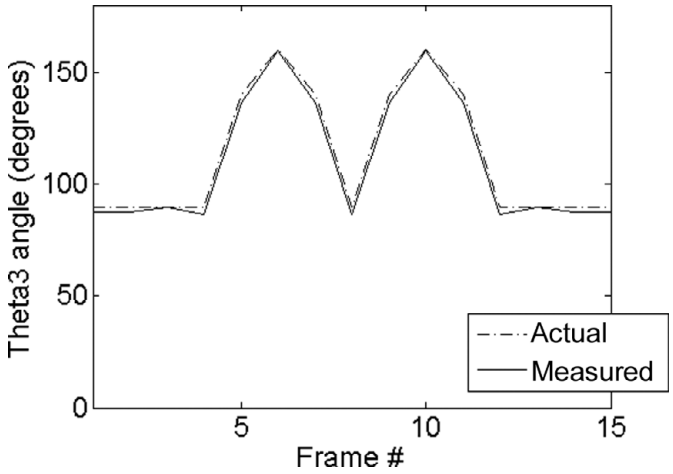

Fig. 6. Actual vs. calculated values for θ_3.

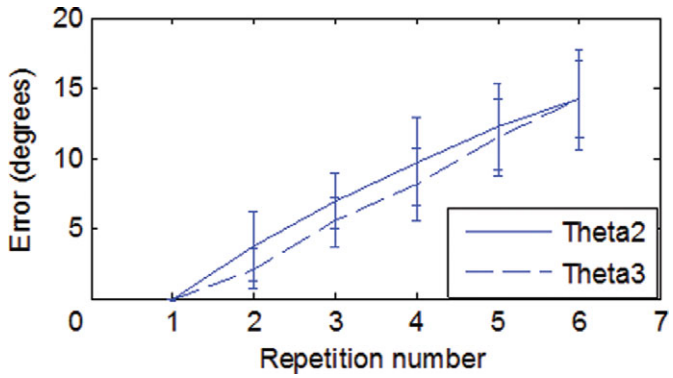

Fig. 7. (Colour online) Degree error per iteration for θ_2 and θ_2 across a subset of sample images.

repeated five times to attempt to capture the true effect of degrative quality loss. It can be seen in Fig. 7 that the error for the detection calculation results is nearly linear. This can be explained by understanding the cause of the visual detection error and simple error propagation. Figure 8 shows an example of iterative error accumulation results for six of the captured data points, including simulated results, for 20 iterations. Clearly, over time even a small amount of error can accumulate to large levels. 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 iterations, the chance for landmark detection error remains the same, as it

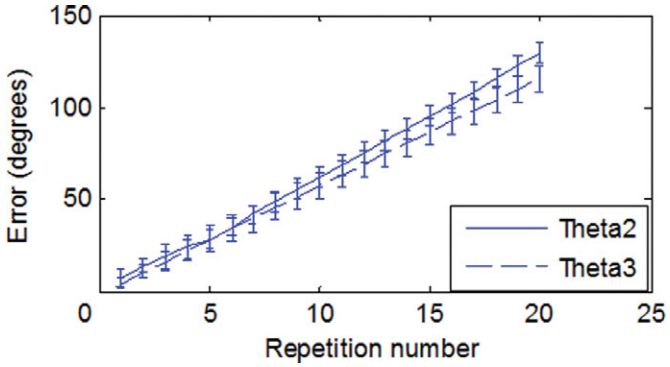

Fig. 8. (Colour online) Projected angle error for up to 20 repetitions.

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.

5.4. 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. Nine behaviors were examined: Waving, Hailing a cab (hail_cab), Cheering, Exercise, Flapping, Weights, and three versions of Curl.

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°, the high-level motion detection system began to falsely identify motions. Because the original experiment was limited to one example plus five repetitions (for convenience), we later simulated additional repetitions by appending new "data" using the same linear error rate as seen in the previously measured iterations. These one-sided error rates are listed in Table I. Values for angle measurements for repetitions past the sixth 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 computed for separate instances of each repetition by adding a random error of +/− 0% to 100% of the average error (delta as in Table I). A demonstration of the overall trend in angle measurements is shown in Fig. 8 (which was created by extending Fig. 7 by simulating double-sided error for 20 repetitions at the measured error rates

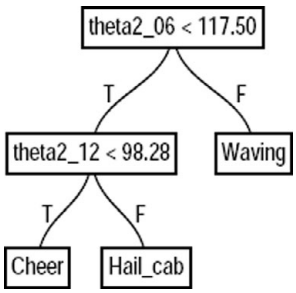

Fig. 9. Classification results for 20 repetitions of three behaviors. Theta2 and Theta3 represent the angles of the upper arm joints, and the _*xx* suffix indicates the frame number of the 15-frame behavior sequence.

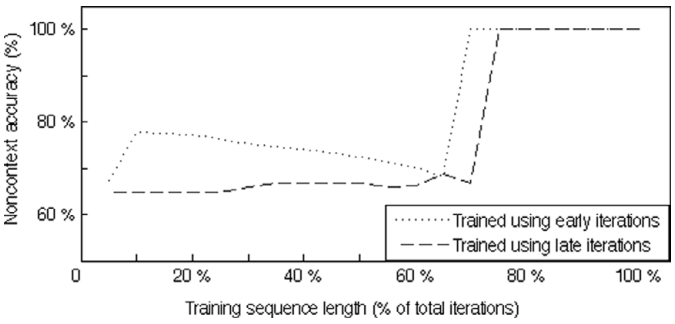

Fig. 10. Classification accuracy for out-of-context test data. The *y*-axis shows test set accuracy for out-of-context data. The *x*-axis shows how many (sequential) iterations were used to train the tree, as a percentage of the 20 iterations performed. The dotted line corresponds to the tree trained with "early" (context) iterations, the dashed line was trained with "late" (context) iterations, and tested on "early" iterations.

for one to six repetitions). It is clear that even with only a linearly increasing error rate, some error values begin to drift toward very high levels. 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 inducer[64] was used to classify the observed angle sequences into specific behaviors. The resulting decision tree is shown in Fig. 9. The attributes shown in the tree are the angles of Θ_2 (theta2) and Θ_3 (theta3). The second number is the frame number in the behavior sequence (which can be used because the behavior sequences are set to known lengths). For example, in the "Waving" behavior sequence, in frame 1, the ideal value for Θ_2 is 0°. This would correspond in the tree to theta2_01 = 0.

When data from all of the behavior repetitions are used to train the tree, the decision tree method was able to classify test set data from every iteration with 100% accuracy. This is excellent considering the high level of error in later behavior repetitions and shows that, when provided data from all contexts, the ITI can identify and use attributes that are resistant to context drift. However, ITI's priority is to choose attributes that result in good performance *now*, rather than attributes that are resistant to drift. Trees induced using only early data do not classify late data well, and vice versa. Section 5.5 shows that our method, OLDEX, can be applied in order to define context regions where ITI performs well despite drift.

5.5. OLDEX: On-line DElimiter for conteXt

This section describes the results for two types of context changes: (1) Context changes due to concept drift, and (2) Context changes due to on-line configuration changes.

5.5.1. Context changes due to concept drift. This section demonstrates a potential use of the system to identify critical points where concept drift becomes a problem. For the sake of clarity, plots are limited to a subset of three behaviors—Waving, Hail_cab, and Cheer.

Why should we be concerned about a slowly drifting context change when Fig. 9 show that a very accurate tree can be induced using the entire set of data? Figure 10 shows that there is a dramatic increase in out-of-context classification accuracy after a certain percentage of training iterations. This suggests that in the presence of accumulating error, there

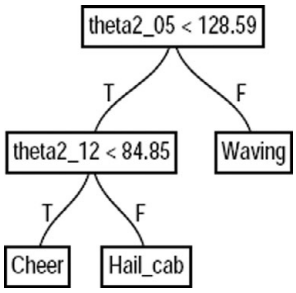

Fig. 11. Decision tree trained using repetitions 0–12 (with 0 being the original input angles).

may be a natural delineation in the training data between "early" and "late" observations. The accuracy improvement occurs after about 66% of the training context is applied for training the classification tree, regardless of whether the "early" context or the "late" context is used for training.

The improvement occurs because the widening error between subsequent iterations eventually forces the inducer to choose error-resistant attributes. This also indicates that the error-resistant attributes are not (initially) the best ones for entropy reduction. By separating the data into context regions where the drift is sufficiently small, we can build trees that perform well in terms of entropy reduction. Misclassification of later data is avoided by using the tree from the current context. To find attributes that both minimize entropy and are resistant to error, we can define contexts where the drift is small enough that the entropy-based inducer works well.

We examined the data assuming a two-context condition, an "early" context and a "late" context, in a subset of the robotic behavior dataset (Waving, Hail_cab, and Cheer). The output decision tree from ITI showed a change in attribute choice between iterations 12 and 13; thus, OLDEX places a context break here. Figures 11 and 12 show the different trees produced when the context is split between iterations 12 and 13. Both have 100% same-context test set accuracy. This location is confirmed by examining Fig. 10, which shows an accuracy improvement after the 12th iteration. OLDEX was able to identify the location of a time-based context shift, without needing to see the entire dataset or isolate a window of data for examination. Thus, it has potential for

Table II. Accuracy for test data from an "original," undamaged robot.

	Test on "original," undamaged robot		Test on damaged robot	
Training context	Model 1 train on original robot	Model 2 train on damaged robot	Model 1 train on original robot	Model 2 train on damaged robot
No robots break down	99.98% +/− 6.67%	n/a	n/a	18.22% +/− 2.18%
One robot breaks down	99.98% +/− 6.67%	67.21% +/− 4.76%	18.22% +/− 2.18%	67.21% +/− 4.76%
Both robots break down	66.77% +/− 3.24%	63.43% +/− 6.69%	63.13% +/− 7.79%	63.43% +/− 6.69%

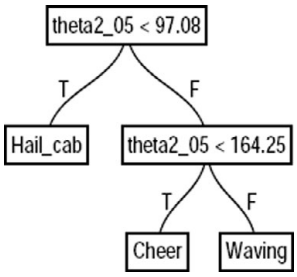

Fig. 12. Decision tree trained using only repetitions 13–20. Note the changes in the branch nodes.

quantitatively identifying a context delimiter in an on-line manner.

To compare our method with SLICE, we included the iteration number as a timestamp and tested the entire dataset off-line in one batch using ITI. The output decision tree did not choose the iteration (time) attribute in any decision nodes. Therefore, SLICE would fail to find a context division for this data, despite the large amount of drift that occurs. However, we argue that a context shift does occur, which is demonstrated by the poor classification accuracy of "late" context data by trees from "early" iterations. The windowed versions of SLICE might be able to define a context shift at the location seen in Fig. 10 (between iterations 12 and 13). OLDEX found this location of interest on-line, immediately after seeing iteration 13. This demonstrates the usefulness of OLDEX for delimiting slowly time-varying contexts for C4.5-type classifiers.

5.5.2. Context changes due to changes in physical configurations. This section demonstrates how OLDEX can be used to create multiple models for context changes due to changing configurations. We simulate a change in physical configuration by "removing" part of a robot after the training has begun. The data for that part is simply deleted. We wish to know whether behavior can be transmitted between two robots, despite dissimilar configurations, by identifying and storing unique configuration models as they arise. All nine behavior sequences were used in this section.

The behavior sequences are defined by a sequentially ordered set of movements. Each behavior depends on (1) the position of a robot part, and (2) the frame number in the sequence corresponding to that position. When we remove a robot part, the definition of the behavior sequence must change. Therefore, we create a model corresponding to the new configuration. Although this model necessarily differs from the original definition of the behavior, it is a close approximation and represents the motions that can actually be performed by the modified robot.

Each instance of a "behavior" consists of a 15-image "video". During the video, the robot moves its arm up and down. The sequence of up and down motions during the 15 images defines the behavior, i.e., during the "waving" behavior, the arm moves completely up and down twice during the sequence. The motion is quantified by extracting the joint angles (as described in the earlier sections). To distinguish the behaviors, we create a model by inducing a decision tree using joint angle data from nine different behavior sequences. To test the model, previously unseen test data are classified using the decision tree. The goodness of the model is quantified by whether the model can correctly identify the behavior from the unseen test data.

To investigate the effects of context changes, we introduced two types of contexts:

(1) Accumulating, unidirectional error in the joint angle.
    The joint overshoots the angle a little during each repetition. Over time, the accumulated error becomes large enough to affect the induced model.
(1) Physical configuration change.
    The loss of an arm segment is simulated by deleting data for the elbow angle (Theta3). The loss occurs midway through the experiment.

The accumulating error context (1) was previously applied for all conditions. Its direct effects are shown in Fig. 10. Now, we show the effects of adding a second error—(2) Physical configuration change. An arm segment was removed virtually from one, or from both robots. Missing limbs were simulated by setting the data to "missing" at the appropriate point in the sequence. When a student robot has more joints than the teacher robot, it uses the angle setting from the last observation of that joint (plus the accumulating error). If the student has fewer joints, it simply ignores the extra data. 10-fold cross-validation was performed. We assume there are only two robots in this training scenario.

For this experiment, 20 repetitions of the original data angles were performed by repetitions between "two" robots (five measured repetitions, 15 simulated). Initially, data from both robots were used to train a single model. Then, a configuration change was simulated by deleting the elbow angle data (Theta3) from one or both robots midway through the training sequence. At this point, a second model was created, trained exclusively using angle data and output classes from the "damaged" robot.

Table II shows the test set accuracy for each configuration-based model. The nine behavior sequences were used to

simulate an on-line configuration change. Model 1 was trained using the original robot configuration. Model 2 was trained using damaged robots. The test set accuracy is shown +/− 1 standard deviation. Table II shows the test set accuracies when the test set comes from an "original" robot, and compares them to accuracy results from a "damaged" robot.

Table II shows that using the correct model ("original" or "damaged") for the data results in the best accuracy. First, the test data are generated by an "original" configuration robot, and the original robot model gives better accuracy whether or not a robot breaks down mid-sequence. Then, the test set containing data from a "damaged" robot is classified more accurately using a "damaged robot" model. These results demonstrate the usefulness of a multiple model approach for classification in scenarios where context changes occur.

6. Conclusion

The purpose of this research was to create a system in which behavior imitation between multiple humanoid robots could be implemented and identified despite changing physical configurations and uncorrected accumulating error. This paper has three important contributions. (1) We propose a new repetition framework for vision-based behavior imitation by a sequence of multiple humanoid robots, (2) we introduce a new on-line decision tree inducer (OLDEX) that can identify context changes and autonomously compensate by creating new, separate context models, and (3) we demonstrate mutual visual-based imitation by humanoid robots, which to our knowledge has never been attempted. In our framework, the robot and camera acted as parts of two unique systems that iteratively captured and determined the motion from a previous robot's motion plan execution. Error accumulation over time was proposed as a model of time-varying context drift. The occurrence of configuration changes was also defined as a context change. The final goal was the transfer and implementation of "high-level motions" or basic behaviors between multiple humanoid robots using visual motion capture.

Although ITI can induce an accurate tree for a drifting context, it does not initially choose attributes that are resistant to drift and error, even when the context varies slowly and linearly. Refining the processing techniques, i.e., using a more powerful low-level landmark detection method, might improve the results for early iterations. However, this would only lengthen the time before classification failure occurred. Our method (OLDEX) demonstrated a simple on-line strategy for quantitatively identifying context regions that were suitable for an entropy-based decision tree inducer. This method can be used to discover on-line changes to the system, even while normal classification procedures are operating. It could also function as an autonomous flag indicating a need for system maintenance or attention.

The ability to autonomously identify context changes within a behavior over time could be an important step toward segmenting unstructured video. By applying a similar process (of observing decision tree changes) to a sequence of behaviors, it may be possible to identify behavior "regions" within a video. This may require a more nonparametric definition of behavior, i.e., defining a behavior as a set of changes to joint angles, rather than specific joint angles at specific times. However, we believe that observing changes in the classifier opens the door to autonomous identification of segmentation points for video sequences.

An important future step of this work would be feedback/reinforcement of behaviors. The system does not have the feedback/reinforcement mechanisms that would assess fitness of a particular configuration for a primitive. That would require some mechanism for making judgments on functionality of behaviors. Although feedback/reinforcement is outside of the scope of this paper, it would certainly increase the usefulness of this research. Once reinforcement can be returned to the system regarding suitability, self-modification of behavior could be implemented. This would likely eliminate drift of primitive definitions and could be used to optimize primitives based on their suitability. With incorporation of reinforcement or feedback, the robots' ability to observe others would give heterogeneous sources for performance solutions, which could lead to emergence of previously unseen, improved solutions.

A few interesting approaches for reinforcement would be examination of the object-state-space of Kruger *et al.*[34] which could handle multiple approaches for achieving a goal behavior, or feedback along the lines of Peters and Schaal's[45] reinforcement learning of operational space parameters. One simple way to incorporate feedback would be to require a robot limb to reach a specific position in order for the data to be incremented into the model. For example, a "press button" behavior could be defined where the robot has to place its hand on a button. Reaching the button would be considered a "success," and the sequence would be incorporated into the model. Sequences that did not produce "success" would be ignored. This would still provide a variety of data for training the model, since there may be different combinations of joint angles, which effectively place the hand on the button.

Other possible future studies may include organizing a flow of repetition, such as interchanges between the humanoid robots, and replaying back previously learned behaviors. Another application 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 without requiring software or cross-platform compatibility.

The applications of humanoid robot–robot imitation may not be self-evident. However, we believe the value of this research is in its contribution to laying groundwork for visual imitation between dissimilar structures, and structures that change unpredictably over time. This contribution extends beyond the limitations of humanoid anatomy structures and extends work for dynamic structures. Learning by imitation has been extensively studied in the robot–human imitation domain, so it may seem simplistic and unnecessary for a robot to imitate another robot. After all, an effective performer of the behavior exists (the human), and so the robot naturally

should imitate the best model. However, this limits robot imitation to behaviors that can be modeled by humans. Human actions are limited by a standard, common anatomy. Robots are not subject to this limitation. They can be given any physically feasible anatomical configuration, and their configurations can be changed over time (intentionally or unintentionally).

Autonomous control of unpredictably changing structures is a nascent and important field. The motion acquisition methods applied are primitive, but acquisition is beyond the scope of this paper. Other practitioners can, and should, apply their own more advanced techniques. Once the data are acquired, then it can be fed into the algorithm presented here.

The paper focuses extensively on the on-line classification method because it is the pivotal part of this learning by imitation procedure. We apply a machine learning method for classification in order to create a learning framework that is independent of the type of system. Thus, our decision tree method can be applied not only to robots, but to any sort of system that changes unpredictably over time.

In conclusion, these studies help to demonstrate the usefulness of an evolutionary learning scheme in time-varying contexts, as well as helping to emphasize points that should be taken into account in order to apply them effectively.

Acknowledgments

The authors would like to thank Dan Couture and Xianhua Jiang for the initial contributions to this study, Josh Bongard for helpful discussions on robot imitation, and the anonymous reviewers for their detailed comments, which improved the paper. This study is partly supported by MED Associates. The second author (Y. Motai) acknowledges the support of NSF Division of Electrical, Communications and Cyber Systems, CAREER Award # 1054333.

References

1. R. Amit and M. Matarić, "Learning Movement Sequences from Demonstration," *Proceedings of the International Conference on Development and Learning (ICDL '02)*, Cambridge, Massachusetts (2002) pp. 203–208.
2. A. Arsenio, "Children, Humanoid Robots and Caregivers," *Proceedings of the 4th International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems Children*, Genoa, Italy (2004) vol. 117, pp. 19–26.
3. C. G. Atkeson, A. W. Moore and S. Schaal, "Locally weighted learning," *Artif. Intell. Rev.* **11**(1), 11–73 (1997).
4. D. Bentivegna and C. G. Atkeson, "Using Primitives in Learning from Observation," *Proceedings of the 1st IEEE-RAS International Conference on Humanoid Robots*, Boston, MA (2000).
5. D. C. Bentivegna and C. G. Atkeson, "Learning from Observation Using Primitives," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'01)* (IEEE, Piscataway, NJ, USA, 2001), Seoul, Korea, vol. 2, pp. 1988–1993.
6. A. Billard and G. Hayes, "DRAMA, a connectionist architecture for control and learning in autonomous robots," *Adapt. Behav.* **7**(1), 35–63 (1999).
7. A. Billard and M. J. Matarić, "Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture," *Robot. Auton. Syst.* **37**(2–3), 145–160 (2001).
8. J. Bongard and R. Pfeifer, "Evolving Complete Agents Using Artificial Ontogeny,". **In:** *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)* (Springer-Verlag, Berlin, 2003) pp. 237–258.
9. J. Bongard, V. Zykov and H. Lipson, "Resilient machines through continuous self-modeling," *Science* **314**(5802), 1118–1121 (2006).
10. Boyd, R. S., 2009, "Robots are narrowing the gap with humans," http://www.mcclatchydc.com/ 226/story/66530.html
11. L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees* (Wadsworth, Belmont, CA, 1984).
12. S. Calinon and A. Billard, "Recognition and Reproduction of Gestures using a Probabilistic Framework combining PCA, ICA and HMM," *Proceedings of the International Conference on Machine Learning (ICML), Bonn, Germany, August 2005* (2005) pp. 105–112.
13. S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Trans. Syst. Man Cybern. B*, **37**(2), 286–298 (2007).
14. F. Cao and B. Shepherd, "MIMIC: A Robot Planning Environment Integrating Real and Simulated Worlds," *Proceedings, IEEE International Symposium on Intelligent Control* (IEEE, Piscataway, NJ, USA, Sep. 25–26, 1989), Albany, NY, USA, pp. 459–464.
15. E. Cole, "AMARSi project could see robots learn from co-workers," Retrieved Mar. 17, 2010, http://www.wired.co.uk/news/archive/2010–03/12/amarsi-project-could-see-robots-learn-from-co-workers.aspx
16. A. K. Dey and G. D. Abowd, "Towards a Better Understanding of Context and Context-Awareness," Technical Report, GIT-GVU-99-22. Georgia Institute of Technology (1999).
17. E. Drumwright and M. J. Matarić, "Generating and Recognizing Free-Space Movements in Humanoid Robots," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, Piscataway, NJ, USA, 2003), Las Vegas, NV, USA, vol. 2, pp. 1672–1678.
18. J. L. Drury, J. Scholtz and H. A. Yanco, "Awareness in Human-Robot Interactions," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*— (IEEE, Piscataway, NJ, USA, 2003), Washington, DC, USA, vol. 1, pp. 912–918.
19. T. Fong, I. Nourbakhsh and K. Dautenhahn, "A Survey of Socially Interactive Robots," *Robotics and Autonomous Systems* **42**(3–4), 143–166 (2003).
20. J. H. Friedman, "A recursive partitioning decision rule for nonparametric classification," *IEEE Trans. Comput.* **26**(4), 404–408 (1977).
21. E. Hamner, R. Gockley, E. Porter and I. Nourbakhsh, "The personal rover project: The comprehensive design of a domestic personal robot," *Robot. Auton. Syst. Special Issue on Socially Interact. Robots* **42**(3–4), 245–258 (2003).
22. I. Haritaoglu, D. Harwood and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 809–830 (2000).
23. M. B. Harries, C. Sammut and K. Horn, "Extracting hidden context," *Mach. Learn.* **32**(2), 101–126 (1998).
24. G. Hulten, L. Spencer and P. Domingos, "Mining Time-Changing Data Streams," *Paper Presented at the Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA (2001) pp. 97–106.
25. E. B. Hunt, J. Marin and P. J. Stone, *Experiments in Induction* (Academic Press, New York, NY, USA, 1966).
26. A. J. Ijspeert, J. Nakanishi, T. Shibata and S. Schaal, "Nonlinear Dynamical Systems for Imitation with Humanoid Robots," *Proceedings of the 2nd IEEE-RAS International Conference*

on Humanoid Robots, Tokyo, Japan, (IEEE, Piscataway, NJ, USA, 2001) pp. 219–226.
27. Y. Inoue, T. Tohge and H. Iba, "Object Transportation by Two Humanoid Robots Using Cooperative Learning," *Proceedings of the 2004 Congress Evolutionary Computation*, Portland, OR, USA (IEEE, Piscataway, NJ, USA, 2004) vol. 1, pp. 1201–1208.
28. O. Khalid, "A unified approach for motion and force control of robot manipulators: the operational space formulation," *IEEE J. Robot. Autom.* **3**(1), 43–53 (1987).
29. B. Kim and G. Lee, "Decision-Tree Based Error Correction for Statistical Phrase Break Prediction in Korean," *Paper presented at the Proceedings of the 18th Conference on Computational linguistics (COLING)* (Morgan Kaufmann Publishers, *Saarbrücken, Germany*, San Francisco, CA, USA, 2000) vol. 2, pp. 1051–1055.
30. V. Klingspor, J. Demiris, and M. Kaiser, "Human-robot-communication and machine learning," *Applied Artificial Intelligence Journal* **11**(7/8), 719–746 (1997).
31. Kondo Kagaku Co., Ltd., Jun. 30, 2008, Retrieved Nov. 7, 2008, http://www.kondo-robot.com/
32. K. Kosuge and T. Oosumi, "Decentralized Control of Multiple Robots Handling an Object," *Proceedings of the IEEE Int. Conf. Intelligent Robots and Systems (IROS '96)*, Osaka, Japan (IEEE, Piscataway, NJ, USA, Nov. 4–8, 1996).
33. H. Kozima and H. Yano, "A Robot that Learns to Communicate with Human Caregivers," *Proceedings of the 1st International Workshop on Epigenetic Robotics* (Lund University Cognitive Studies, Lund, Sweden, Lund, Sweden, 2001).
34. V. Kruger, D. Herzog,S. Baby, A. Ude and D. Kragic, "Learning actions from observations," *IEEE Robot. Autom. Mag.* **17**(2), 30–43 (2010).
35. J.-S. Liu, T.-C. Liang and Y.-A. Lin, "Realization of a ball passing strategy for a robot soccer game: A case study of integrated planning and control," *Robotica* **22**(3), 329–338 (2004).
36. W.-Y. Loh and Y.-S. Shih, "Split selection methods for classification trees," *Statistica Sinica* **7**, 815–840 (1997).
37. A. Martinoli, A. J. Ijspeert and L. M. Gambardella, "A Probabilistic Model for Understanding and Comparing Collective Aggregation Mechanisms," *Proceedings of the 5th European Conference on Advances in Artificial Life* (Springer-Verlag, Berlin/Heidelberg, 1999), Lausanne, Switzerland, vol. 1674, pp. 575–584.
38. M. J. Matarić, "Reinforcement learning in the multi-robot domain," *Auton. Robots* **4**(1), 73–83 (1997).
39. M. J. Matarić, "Sensory-Motor Primitives as a Basis for Imitation: Linking Perception to Action and Biology to Robotics," **In:** *Imitation in Animals and Artifacts* (K. Dautenhahn and C. L. Nehaniv, eds.), (MIT Press, Cambridge, MA, 2002) pp. 391–422.
40. R. A. McCallum, "Hidden state and reinforcement learning with instance-based state identification," *IEEE Trans. Syst. Man Cybern.* **26**(3), 464–473 (1996).
41. Motion Analysis, Inc., Retrieved Nov. 7, 2008, http://www.motionanalysis.com
42. M. N. Nicolescu and M. J. Matarić, "Natural Methods for Robot Task Learning: Instructive Demonstrations, Generalization and Practice," *in Proceedings Second International Joint Conference on Autonomous Agents and Multi-Agent Systems* pages 241–248, Melbourne, Australia, July 14–18, 2003.
43. A.P.A.S, Ariel Dynamics, Jun. 30, 2008, Retrieved Nov. 7, 2008, http://www.arielnet.com/
44. G. A. S. Pereira, V. Kumar, J. R. Spletzer, C. J. Taylor and M. F. M. Campos, "Cooperative Transport of Planar Objects by Multiple Mobile Robots Using Object Closure," **In:** *Experimental Robotics VIII* (Springer, Berlin/Heidelberg, 2003) vol. 5, pp. 287–296.
45. J. Peters and S. Schaal, "Policy Gradient Methods for Robotics," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, (2006) pp. 2219–2225.
46. D. Perzanowski, A. C. Schultz, W. Adams, E. Marsh and M. Bugajska, "Building a multimodal human-robot interface," *Intell. Syst.* **16**(1), 16–21 (2001).
47. N. Phillips, 2010, "The robots are cutting in on our dance moves," http://www.smh.com.au/technology/technology-news/the-robots-are-cutting-in-on-our-dance-moves-20100827–13vzn.html
48. J. Piaget, *Play, Dreams, and Imitation in Childhood* (Gattegno, C. and Hodgson, F. M., Trans.) (Norton, New York, NY, USA, 1962 (translation), 1945 (French)).
49. N. S. Pollard, J. K. Hodgins, M. J. Riley and C. G. Atkeson, "Adapting Human Motion for the Control of a Humanoid Robot," *Proceedings of the IEEE International Conference on Robotics and Automation* (IEEE, Piscataway, NJ, USA, 2002) Washington, DC, USA, vol. 2, pp. 1390–1397.
50. J. R. Quinlan, "Discovering Rules by Induction from Large Collections of Examples," **In:** *Expert Systems in the Micro-electronic Age* (D. Michie, ed.) (Edinburgh University Press, Edinburgh, UK, 1979).
51. J. R. Quinlan, "Induction of decision trees," *Mach. Learn.* **1**(1), 81–106 (1986).
52. J. R. Quinlan, *C4.5: Programs for Machine Learning* (Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA, 1993).
53. R. A. Rescorla, "Probability of shock in the presence and absence of CS in fear conditioning," *J. Comp. Physiol. Psychol.* **66**, 1–5 (1968).
54. S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends Cogn. Sci.* **3**(6), 233–242 (1999).
55. J. C. Schlimmer and D. Fisher, "A Case Study of Incremental Concept Induction," *Proceedings of the 5th National Conference on Artificial Intelligence* (Morgan Kaufmann, Philadelphia, PA, USA, 1986) Philadelphia, PA, USA, Vol. 1, pp. 495–501.
56. J. C. Schlimmer and R. H. Granger, "Incremental learning from noisy data," *Mach. Learn.* **1**(3), 317–354 (1986).
57. G. H. Shah Hamzei, D. J. Mulvaney and I. P. W. Sillitoe, "Batch-Mode Decision Tree Learning Applied to Intelligent Reactive Robot Control," *Proceedings of the 6th International Conference on Emerging Technologies and Factory Automation (ETFA '97)* Los Angeles, CA, USA, (IEEE, Piscataway, NJ, USA, 1997) pp. 416–420.
58. C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 747–757 (2000).
59. L. Steels and P. Vogt, "Grounding Adaptive Language Games in Robotic Agents," In *Proceedings of the 4th European Conference on Artificial Life* Brighton, UK, (MIT Press, Cambridge, MA, USA/London, 1997) pp. 474–482.
60. J. Tani, R. Nishimoto, J. Namikawa, and M. Ito, "Codevelopmental learning between human and humanoid robot using a dynamic neural-network model", *IEEE Trans. on Syst.* Man and Cybern. Part B-Cybernetics **38**(1), pp. 43–59, 2008.
61. K. C. Tan, Y. J. Chen, K. K. Tan and T. H. Lee, "Task-oriented developmental learning for humanoid robots," *IEEE Trans. Ind. Electron.* **52**(3), 906–914 (2005).
62. P. Utgoff, Mar. 23, 2001, "Incremental tree induction," Retrieved Nov. 7, 2008, http://www-lrn.cs.umass.edu/iti/index.html
63. P. E. Utgoff, "Incremental induction of decision trees," *Mach. Learn.* **4**(2), 161–186 (1989).
64. P. E. Utgoff, N. C. Berkman and J. A. Clouse, "Decision tree induction based on efficient tree restructuring," *Mach. Learn.* **29**(1), 5–44 (1997).

65. Vicon Motion Systems, Oxford Metrics Ltd., Retrieved Nov. 7, 2008, http://www.vicon.com
66. S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt and S. Schaal, "Statistical learning for humanoid robots," *Auton. Robots* **12**(1), 55–69 (2002).
67. G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.* **23**(1), 69–101 (1996).
68. C. R. Wren, A. Azarbayejani, T. Darrell and A. P. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.* **19**(7), 780–785 (1997).
69. H. A. Yanco, "Synthetic Robot Language Development," In *Proceedings of the Twelfth National Conference on Artificial Intelligence.* Seattle, Washington, USA. AAAI Press/The MIT Press, 1994. p. 1500.
70. Y. Yokokohji, Y. Kitaoka and T.Yoshikawa, "Motion capture from demonstrator's viewpoint and its application to robot teaching," *J. Robot. Syst.* **22**(2), 87–97 (2005).