In *AAMAS 2010 Workshop on Agents Learning Interactively from Human Teachers (ALIHT)*,
Toronto, Canada, May 2010.

Beyond Teleoperation: Exploiting Human Motor Skills with MARIOnET

Adam Setapen
Dept. of Computer Science
University of Texas at Austin
asetapen@cs.utexas.edu

Michael Quinlan
Dept. of Computer Science
University of Texas at Austin
mquinlan@cs.utexas.edu

Peter Stone
Dept. of Computer Science
University of Texas at Austin
pstone@cs.utexas.edu

ABSTRACT

Although machine learning has improved the rate and accuracy at which robots are able to learn, there still exist tasks for which humans can improve performance significantly faster and more robustly than computers. While some ongoing work considers the role of human reinforcement in intelligent algorithms, the burden of learning is often placed solely on the computer. These approaches neglect the expressive capabilities of humans, especially regarding our ability to quickly refine motor skills. In this paper, we propose a general framework for Motion Acquisition for Robots through Iterative Online Evaluative Training (MARIOnET). Our novel paradigm centers around a human in a motion-capture laboratory that "puppets" a robot in realtime. This mechanism allows for rapid motion development for different robots, with a training process that provides a natural human interface and requires no technical knowledge. Fully implemented and tested on two robotic platforms (one quadruped and one biped), this paper demonstrates that MARIOnET is a viable way to directly transfer human motion skills to robots.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics—*Operator interfaces, Kinematics and dynamics*

General Terms

Human Factors, Design, Experimentation

Keywords

Human-robot/agent interaction, Agent development techniques, tools and environments

1. INTRODUCTION

As robots become more commonplace, the tools to facilitate knowledge transfer from human to robot will be vital, especially for non-technical users. Therefore, when designing autonomous robots that interact with humans, not only is it important to leverage advances in machine learning, but it is also useful to have the tools in place to enable direct transfer of knowledge between man and machine. We introduce such a tool for enabling a human to teach motion capabilities to a robot.

Specifically, this paper describes a direct and real-time interface between a human in a motion-capture suit and a robot. In our framework, the learning happens exclusively by the human - not the robot. Our approach exploits the rate at which humans are able to learn and refine fine-motor skills [19, 14]. Called *MARIOnET*, **M**otion **A**cquisition in **R**obots through **I**terative **On**line **E**valuative **T**raining, the interface has been implemented on two robots - one quadruped and one humanoid. As the name indicates, *MARIOnET* is a form of iterative online evaluative training. The human performs a motion and the robot mimics in realtime. The human evaluates the robot's performance, and repeats the motion accounting for any errors perceived in the robot's previous actions. This loop is continued until a sufficient motion sequence is obtained.

Our results indicate that humans are able to quickly improve a robot's performance of a task requiring fine-motor skills.The primary contribution of this paper is a new paradigm for directly encoding motion sequences from a human to a robot. One motivation of our approach is to develop an efficient method for generating cyclical motion sequences. We empirically evaluate the rate at which human subjects learn to exploit a direct robot mapping, and demonstrate that *MARIOnET* is a powerful way to harness the cognitive flexibility of humans for quickly training robots.

The remainder of the paper is organized as follows. Section 2 describes our motivation and provides background information necessary to understand the technical details of our approach. Section 3 provides a detailed account of the implementation and training process of *MARIOnET*. Section 4 outlines the hardware used in our evaluation and describes our experimental results. Section 5 discusses related work, and finally Section 6 presents our conclusions and possible future work.

2. MOTIVATION AND BACKGROUND

This section provides detailed motivation for our work, briefly situates *MARIOnET* within the learning agents literature, and describes the underlying technology upon which *MARIOnET* is built.

2.1 Motivation

In this paper, we examine a new approach towards robot motion acquisition through an innovative training paradigm that exploits a system finely tuned by thousands of years of biological evolution: the human body. While the process by which humans are able to learn exceptionally quickly is not yet fully understood, work being done on the neurological basis of learning is steadily shedding light on how we rapidly acquire and apply new knowledge [13]. Recent breakthroughs in behavioral motor control have enhanced our understanding of the human brain and illustrate how remarkable our innate capacity for delicate motor control is [19]. Muellbacher et al. report that given a 60-minute training period, human subjects can rapidly optimize performance of a complex task involving fine motor control [14].

The high-level motivation for *MARIOnET* is that a real-time mapping from a human to a robot will serve as a convenient interface for quickly and systematically training efficient motion sequences. While there is certainly a difference in the natural dynamics of robots and Homo sapiens, it is our belief that people's ability to quickly hone fine motor skills can be exploited to rapidly train diverse robot motions. Even if the mapping from human coordinates to robot coordinates is not exact, we hypothesize that humans will be able to rapidly learn to correct for any inconsistencies. Additionally, the prospect of mapping any human limb to any robot limb allows for a flexible training process (e.g., mapping human arms to robot legs).

Our long-term vision is to use *MARIOnET* to train robust motions for robots such as walking, running, and kicking. This could be useful in many domains, such as robot soccer where most people create these behaviors by hand or via extensive learning experiments using constrained parameterizations, causing a lot of wear and tear on the robots [10]. Generally, programming specialized robot motions requires a significant amount of coding, which is not possible for most people, and is not necessary when using *MARIOnET*.

MARIOnET uses real-time mimicking on the part of the robot because we believe it is essential that the human subject is able to quickly evaluate the actions of the robot in order to refine its movement in subsequent training. For example, imagine the following situation in attempting to "puppet" a humanoid robot: the human subject realizes that when training a biped robot to walk, the robot frequently loses balance and topples forward. The human can try different things to correct this — lengthening of stride, reduction of knee-bend, etc., all in real time while watching the robot. Although *MARIOnET* uses teleoperation to interact with the robot, it represents a new methodology for knowledge transfer. The central hypothesis of this work is that humans are skilled enough at fine motor control, that minor nuances essential for maximizing performance in robot locomotion (otherwise discoverable only through computationally expensive and time-consuming exhaustive methods) may be found with significantly less effort. Furthermore, since the training process proposed by *MARIOnET* provides an intuitive interface requiring no technical knowledge, our approach facilitates the direct transfer of motor knowledge from human to robot by any non-technical user.

2.2 ML and HRI

Machine learning (ML), the study of algorithms that improve automatically through experience, has drastically improved the rate at which robots can learn. Recently, machine learning algorithms have seen great success in training robots to move quickly and efficiently, and there are numerous case studies in which ML has been used for on-line and off-line performance improvement in multi-agent autonomous robot environments [10, 16].

Human-robot interaction (HRI) examines the principles and methods by which robots and humans can naturally communicate and interact. As robots become more immersed in our everyday lives, the ability for non-technical users to program, train, and interact with these machines will be vital. Thus, any viable framework for human-robot interaction should require very little technical knowledge to use. Additionally, HRI systems should aim to make the method of communication between robot and human as natural as possible, namely by providing a convenient interface for the human [1]. Steinfeld et al. emphasize the need for any viable human-robot interface to cater to non-technical users, and state that when testing any interface it is "critical to recruit subjects having a broad range of knowledge, experience, and expertise" [20].

2.3 Robot Locomotion

Historically robot motion has been written by experts, and falls into two main categories; open-loop and closed-loop solutions. In both approaches the the limbs are roughly performing trajectory following, where the trajectory is either pre-calculated (open-loop) or is calculated based on sensors and dynamics (closed-loop). Both approaches require significant technical knowledge, considerable development time and neither are suitable for non-technical users.

While machine learning has been applied to learn/optimize these trajectories [18, 22, 11] there is still a large amount of code required to initially define the motion sequences, placing a significant burden on the original robot programmer.

Inverse kinematics (IK) encompasses the task of determining the joint angles that will move a robot's limb to a desired end-effector position given in XYZ coordinates (relative to the robot's body). Two main methods exist for calculating inverse kinematics: a) Jacobian based iterative approaches [5] and b) Direct trigonometric calculation. In the work presented in this paper both techniques are used.

On the humanoid robot used in this paper, each leg has six degrees of freedom. As a result there exist many solutions to the IK calculation. That is, multiple configurations of the joints can result in the same end-effector position. For that reason we use a Jacobian approach to based on a Denavit-Hartenberg [8] representation of the limbs. This approach solves for the smallest set of joint movements that result in the end-effector being within ϵ of the desired location. The advantage of this approach is that the robot generally does a good job of reaching any possible location, even getting as close as possible to impossible ones. The disadvantage is that often multiple iterations over the inverse kinematics solver are required, which can be computationally expensive.

On the quadruped robot we use, each leg has only three degrees of freedom, meaning that in most locations there exists only one unique solution for the IK calculation. For that reason a direct trigonometric approach is used, in which we can accurately determine the three required angles. This approach is extremely fast to calculate and is ideal for many robots with relatively low processing power. However, at a few select locations two solutions exists and the robot can occasionally oscillate between these solutions. This is generally not a problem as these limb positions are rarely needed by the robot.

2.4 Motion Capture

Motion capture is a technique of recording the detailed movements of a performer, thereby capturing a digital model of their movements. Motion capture systems may be passive, where a performer wears reflective materials and light is generated near the camera, or active, where various LEDs on the subject's body emit light which is detected by surrounding cameras. Typically, the LEDs are strobed on and off very quickly in pre-defined groups, which enables realtime marker disambiguation by the cameras. State-of-the-art active motion capture systems allow for precise representations of human poses, resulting in a complete digital representation in Cartesian space.

In this work we use a PhaseSpace IMPULSE active-marker motion capture system that employs 16 high-sensitivity 12.6 megapixel cameras positioned overhead surrounding a 20 by 20 foot tracking area. A human subject wears a black virtual-reality body suit, on which 36 LED markers are strategically placed. With a sample rate of 480 Hz and a latency of less than 10ms, the PhaseSpace IMPULSE system is a fast and accurate way of capturing even the most subtle of human movements.

3. MARIONET

This section describes the *MARIOnET* framework. Section 3.1 describes the implementation details of *MARIOnET*, and section 3.2 describes the *MARIOnET* user interface and training methodology.

3.1 Implementation

MARIOnET has been implemented in two distinct modules: a) a C++ framework with a custom client to connect to a motion-capture server and b) a generalized motion module that is directly implemented on each robot. A fully-functional graphical user interface (GUI) has also been developed that facilitates training. The motion capture data is down-sampled and commands can be sent to the robot at 8 - 30 Hz.

We represent each human limb as a vector of points that can be initialized to a "neutral" position. In this way, we can precisely represent any human pose by relating the current pose to a neutral position. The difference between these vectors is now transformed to a coordinate system appropriate for a particular robot, and a resulting set of robot joint positions is generated by calculating a solution to inverse kinematics. The control flow of our interface can be seen in Figure 1. An initial configuration procedure correlates the bounds of each human subject to the bounds of the robot, and captures a neutral human pose.

The main *MARIOnET* algorithm (Algorithm 1) will now be described. Once initialized, the client enters the main loop and captures the markers from the motion capture server, decoding each point to a body part based on unique marker IDs (line 2). The decoded packet is then transformed from the absolute coordinate system of the motion capture system to a relative coordinate system appropriate for a robot (line 5). This transformation is accomplished by calculating a forward-facing vector orthogonal to the plane created by the human's torso, and rotating every point in the pose accordingly. These vectors, now in a relative coordinate system, are scaled down to the robot's size by considering the subject's body size in conjunction with the robot's physical bounds (line 6).

After scaling to the appropriate robot coordinate system, a mapping is applied from human-limb to robot-limb (line 7). It is possible to map limbs one-to-one (for example when fully controlling a humanoid robot), or one-to-many. For example, a "trot" gait can be generated for quadrupeds by mapping the human's left arm to the robot's front left and back right legs, and the human's right arm to the robot's front right and back left legs. The user can select different mappings through the GUI without the need to recompile any code. The GUI also includes realtime interactive sliders

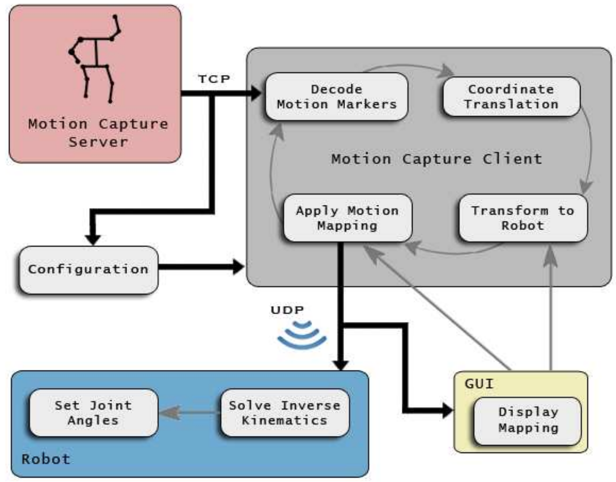


Figure 1: Control flow of the *MARIOnET* Interface

for scaling outgoing robot coordinates. The sliders allow independent control of the x, y, and z values for both arms and legs, which are applied after limb mapping (line 8).

During training, it is often useful to have a "looped" motion sequence. For example, the human could take two steps and wish the robot to repeat this sequence over and over, resulting in a continuous gait. To facilitate a natural human interface, we have implemented hand-gesture recognition to control the looping state of the robot. Whenever the human touches his thumb and pinky fingers together, the robot changes its looping state. There are three looping states: live capture that is not being recorded, live capture that is being recorded for looping, and loop playback. Every time a hand-gesture by the human is detected, *MARIOnET* updates an internal looping signal. All encountered loops are stored, and can be replayed using the GUI or saved directly as a sequence of joint angles on the robot that can be called from any high-level behavioral code to reproduce the looped motion. The lines of Algorithm 1 that deal with looping are 3-4, 9-10, and 13-15.

Input: $config$ (Individual's Configuration), $mapping$ (Desired mapping)

```
1  while TRUE do
2      pose = getMarkers();
3      loopState = handGestureDetected(pose);
4      if loopState != LOOPING then
5          relPose = absoluteToRelative(pose);
6          robotPose = transformToRobot(config);
7          robotPose.applyMapping(mapping);
8          robotPose.scale(GUI.scalars());
9      else
10         (relPose, robotPose) = loop.nextFrame();
11     end
12     sendToRobot(robotPose);
13     if loopState == RECORDING then
14         loop.add(relPose, robotPose);
15     end
16 end
```

Algorithm 1: Main motion-capture algorithm

At this stage, the *MARIOnET* client has a complete representation of the human's body that is scaled down to the robot's coordinate system and altered to represent a particular limb mapping. This information is sent to the robot wirelessly in the form of a packet (line 12), and the algorithm returns to the start of the main loop.

We now turn our attention to the robot motion algorithm, which must be implemented for each model of robot that *MARIOnET* wishes to communicate with. Pseudocode representing a generic robot motion algorithm can be found in Algorithm 2. Every time the robot receives a packet (line 3), it simply calculates a solution to inverse kinematics for each limb (lines 4,5), sets corresponding joint angles using interpolation (line 6), and conveys its current loop state to the user through speech and LED indicators (line 8).

```
1 setPose(INITIAL-POSE);
2 while TRUE do
3     robotPose = getLatestCommand();
4     foreach Effector e ∈ robotPose do
5         [Θ⃗] = solveInvKin(e);
6         interpolateJoints(e, Θ⃗);
7     end
8     conveyLoopState();
9 end
```

Algorithm 2: Robot motion control algorithm

3.2 Training and Interface

The *MARIOnET* GUI allows viewing of both human and robot kinematics. The GUI supports realtime tuning of mapping scalars, and provides a mechanism for recording motion sequences for later use. For example, if the human notices that the robot's arms are always "too close" to the front of its body, the user can simply increase the x-direction scalar of the arms through the GUI.

The training process works best with two people: one controlling the *MARIOnET* GUI and one in the motion-capture suit. The first step in training is creating a configuration file for the human, which is generated by prompting the trainer to successively place their arms at their sides, fully extended forward, fully extended to the sides, and fully extended upward. This process initializes mapping scalars which correlate to the corresponding physical bounds of the robot. The first author of this paper controlled the GUI for all of the training sessions. After initialization, the human and robot are synchronized and live motion-capture data is sent to the robot. The training process is as follows: the human performs a motion, which can be seen through realtime mimicking by the robot. The human evaluates the robot's performance, and repeats the motion accounting for perceived error in the robot's previous actions. This loop is continued until a satisfactory motion sequence is obtained.

4. EXPERIMENTS AND RESULTS

In this section, we describe various experiments that evaluate the effectiveness of our approach. First, we outline the robot hardware used in the current implementation of *MARIOnET*. Then, we analyze the use of *MARIOnET* for an episodic closed-loop task using the Nao. Finally, we show that our approach is also useful for capturing cyclical open-loop motions, such as walking, using the AIBO. This section primarily evaluates the ease of the training interface and assesses the ability of a human to quickly improve at a task involving fine-motor control.

4.1 Experimental Robot Platforms

4.1.1 *Sony AIBO*

The AIBO is a sophisticated quadruped robot that was mass produced by Sony from 1999 to 2006 (see Figure 3). The ERS-7 model (used in these experiments) has an internal 64-bit RISC processor with a clock speed of 576MHz. The robot has 20 degrees of freedom: 3 in the head, 1 in each ear, 1 in the chin, 3 in each leg, and 2 in the tail. The robot also contains a 802.11 wireless card that enables external communication.

4.1.2 *Aldebaran Nao*

We have also implemented *MARIOnET* on a humanoid robot called the Nao (see Figure 2). Developed by Aldebaran Robotics, the Nao recently replaced the AIBO as the robot for the RoboCup Standard Platform League. The Nao contains an AMD Geode 500Mhz processor, 256MB of memory, and includes 802.11 wireless capabilities. Measuring at 23 inches and just under 9.6 pounds, the The Nao has 21 degrees of freedom and body proportions similar to that of a human. Each foot of the robot contains four force-sensitive resistors, and the Nao houses an integrated inertial measurement unit with a two-axis gyrometer and three-axis accelerometer.

4.2 Humanoid Tasks

Eight volunteers served as test subjects, and each subject completed a 45 to 90 minute interactive training session with the Nao. Our test subjects consisted of three technical users and five non-technical users.

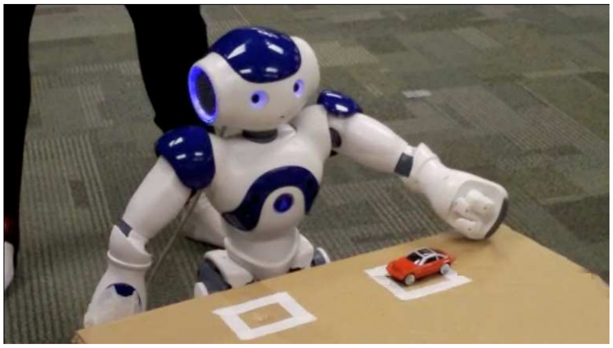
The setup of our episodic task, Car-Park, can be seen in Figure 2(a). The robot stands in front of a surface with two distinct boxes — a source and a sink. The human stands behind the robot and attempts to guide the robot to move a toy car from the source to the sink. The robot starts with both arms at its sides, and the task is completed when all of the car's wheels reside inside the bounds of the sink. If the car is knocked off the surface, the subject is given a three second penalty.

The test subjects performed 60 iterations of Car-Park. For the first 10 episodes, the average time to completion was 28.5 seconds - for the last 10 episodes the average was 6.8 seconds. As can be seen in Figure 2(b) the learning curve representing elapsed time to complete Car-Park decreases significantly over 60 iterations. The entire training session took less than 1 hour, and the subjects decreased their average completion time by a factor greater than 4. This experiment helps verify our hypothesis that humans can quickly learn to control robots via the *MARIOnET* interface.

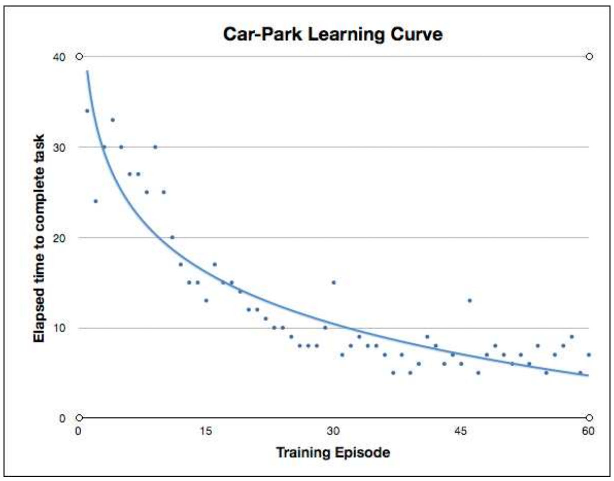
The solution for Car-Park that every user eventually converged on was to use both arms - one to nudge the car and the other to stop it at the correct location. This coordinated sequence is the type of motion that might have taken a standard ML algorithm a long time to find, and would certainly require significant exploration of the state space.

4.3 Quadruped Locomotion

While the Car-Park experiment is a closed loop control task in which the human continually controls the robot, we envision *MARIOnET*'s main usefulness being in generating open loop control sequences such as periodic gaits. Due to the current fragility of the Nao robots, we limited our locomotion tests to the quadruped platform.



(a) Experiment Setup



(b) Results

Figure 2: Experimental setup of Car-Park and average learning curve of 60 iterations

Six volunteers served as test subjects in evaluating the effectiveness of *MARIOnET* for quadruped locomotion, again consisting of both technical and non-technical users (3 technical, 3 non-technical). Due to physical limitations of the body suit, a "crawling" motion by the human was infeasible, because marker visibility on the front of the body suit is critical for coordinate transformations and limb detection. Therefore, we flipped the problem, and the human, upside-down. Each subject laid on his back, and each human limb was mapped to the corresponding robot effector. The task was to get the AIBO to walk one meter.

Four of the test subjects were able to get the robot to walk one meter at least once during a 20 minute training period. The training position with the human on his back is somewhat strenuous, and the other two subjects eventually gave up. However, the four successful subjects exhibited dramatic improvement in walk speed over the course of their sessions. Generally, the subjects had a "moment of clarity", in which they found the correct general trajectory to achieve robot stability. After finding this stable region, the users simply adjusted various aspects of their motion and evaluated any changes on the robot. A learning curve consisting of eight iterations can be seen in Figure 4. While half of the individuals successfully made use of the looping mechanism, the other half preferred to control the robot in real-time for the duration of the task.

Interestingly, the two users unable to produce a walk were

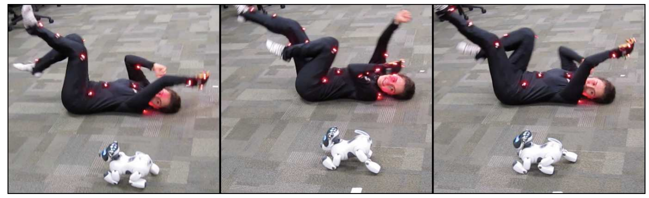


Figure 3: An example training session using the Sony AIBO

both technically inclined (computer scientists). All non-technical subjects achieved a steep learning curve, indicating that technical expertise is not needed to use *MARIOnET*. The fastest looped walk achieved a velocity of 18.8 cm/s, and the subject had only trained for 17 minutes before achieving this speed. To put this number in context, some of the fastest AIBO walks found through optimization algorithms are in excess of 34 cm/s[17], while the standard walk Sony includes with the AIBO is 3.2cm/s. However, most parameter optimization techniques start with a decent hand-coded walk, while *MARIOnET* starts from scratch. It should be noted that the output of the *MARIOnET* learning could be used as the starting point for these optimizations.

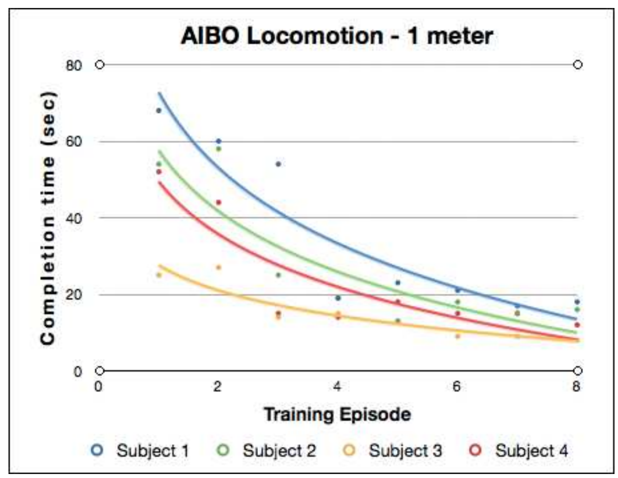


Figure 4: Learning curves of 8 iterations by the four successful subjects

The first time a subject controls a robot, it takes approximately 1 minute to tune the interactive scalars to appropriate values, which correlates a comfortable human pose to a stable robot pose. A video illustrating both robots in action can be found at
`www.cs.utexas.edu/~AustinVilla/?p=research/marionet`.

5. RELATED WORK

Learning from demonstration (LfD, or imitation learning) is a process in which a robot attempts to learn a task by observing a demonstration, typically performed by a human. LfD is a promising way of transferring knowledge from agent to agent, and work by Dautenhahn and Nehaniv illustrates how many animals use this technique to learn new skills [7]. A good deal of recent work in LfD indicates that using human feedback as a reward signal to a reinforcement learning or policy-search algorithm can significantly improve learning speed [21, 9, 6, 2]. These studies illustrate the importance of harnessing human-robot interfaces in order to "design algorithms that support how people want to teach and simultaneously improve the robot's learning behavior" [21]. Thomaz and Breazeal coin this paradigm "Socially guided machine learning", where the benefits of machine learning are combined with the intuitive knowledge of humans.

Breazeal and Scassellati posit that there are four integral questions to consider when designing a system that uses learning from demonstration [4]: a) How does the robot know when to imitate? b) How does the robot know what to imitate? c) How does the robot map observed action into behavior? d) How does the robot evaluate its behavior, correct errors, and recognize when it has achieved its goal?

Our system bypasses the first two questions, as the robot imitates the human in real-time. The robot maps observed actions into behaviors using a deterministic scaling function that can be augmented by the user. Finally, while our robot does not currently evaluate its own behavior, we touch on this question in Section 6.

Although motion-capture data has been harnessed to improve robot locomotion, to the best of our knowledge, no real-time human-robot interface using motion-capture has ever been utilized in the way *MARIOnET* proposes. Recent work by Kulic, Takano, and Nakamura introduced a system using incremental learning of "human motion pattern primitives" by observation of motion-capture data [12]. Additionally, Nakanishi et al. have presented a framework for learning bipedal locomotion using dynamical movement primitives based on non-linear oscillators, using motion-capture data as input [15]. While these approaches are based on a similar motivation of using human motion to train robots, *MARIOnET*'s real-time interface provides a *direct* route of controlling the pose of a robot.

6. CONCLUSIONS AND FUTURE WORK

While the similarities of human movement and robot locomotion have been investigated [3], our idea of exploiting human motor skills for efficient learning of robot locomotion takes a completely new approach. We control the motion of a robot not by modeling its dynamics or tweaking parameters of a machine learning algorithm, but by taking advantage of the most finely-tuned and sophisticated control mechanism known to man: himself.

As more robots appear with complex body dynamics, it is vital that interaction is possible for all types of users, both technical and non-technical. However, it is very difficult to systematically construct motion controllers that exploit the specific properties of a robot, even for a roboticist. Our experiments suggest that all types of subjects are able to successfully use *MARIOnET*, as the non-technical users were able to intuitively grasp the interface. This approach allows the layman to precisely develop specialized robot motions.

In this first specification of *MARIOnET*, we have laid the groundwork for much future work. As mentioned earlier, *MARIOnET* abstracts the task of learning away from the robot and places this burden on the human. Although our results indicate that this approach is viable, a more robust set of problems could be approached and optimized if the robot and human learned in harmony. Three of the four "integral questions" for LfD proposed by Breazeal and Scassellati [4] are naturally answered by *MARIOnET*, while the fourth requires the robot to reason about its actions. Using the effective combination of human reinforcement and machine learning, we plan to address this important question in future work.

7. ACKNOWLEDGMENTS

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-0615104 and IIS-0917122), ONR (N00014-09-1-0658), DARPA (FA8650-08-C-7812), and the Federal Highway Administration (DTFH61-07-H-00030).

8. REFERENCES

[1] Hri '09: Proc. of the 4th acm/ieee intl. conf. on human robot interaction, 2009.

[2] C. Atkeson and S. Schaal. Robot learning from demonstration. In *Machine learning: Proc. 14th Int. Conf. (ICML '97)*, pages 12–20, 1997.

[3] C. Azevedo, B. Espiau, B. Amblard, and C. Assaiante. Bipedal locomotion: toward unified concepts in robotics and neuroscience. *Biological Cybernetics*, 96(2):209–228, 2007.

[4] C. Breazeal and B. Scassellati. *Challenges in building robots that imitate people*. Defense Tech. Information Center, 2000.

[5] S. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse, and damped least squares methods. Technical report, UCSD, 2004.

[6] S. Chernova and M. Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34:1–25, 2009.

[7] K. Dautenhahn and C. Nehaniv. *Imitation in animals and artifacts*. MIT Press Cambridge, MA, USA, 2002.

[8] J. Denavit and R. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 22(2):215–221, 1955.

[9] W. B. Knox and P. Stone. TAMER: Training an Agent Manually via Evaluative Reinforcement. In *IEEE 7th Intl. Conf. on Development and Learning*, August 2008.

[10] N. Kohl and P. Stone. Machine learning for fast quadrupedal locomotion. In *The Nineteenth National Conf. on Artificial Intelligence*, pages 611–616, July 2004.

[11] J. Z. Kolter, P. Abbeel, and A. Y. Ng. Hierarchical apprenticeship learning with application to quadruped locomotion. In *NIPS*, 2007.

[12] D. Kulic, W. Takano, and Y. Nakamura. Combining automated on-line segmentation and incremental clustering for whole body motions. In *Robotics and Automation, 2008. ICRA 2008. IEEE Intl. Conf. on*, pages 2591–2598, 2008.

[13] A. Lawson. *The neurological basis of learning, development and discovery: implications for science and mathematics instruction*. Springer, 2003.

[14] W. Muellbacher, U. Ziemann, B. Boroojerdi, L. Cohen, and M. Hallett. Role of the human motor cortex in rapid motor learning. *Experimental Brain Research*, 136(4):431–438, 2001.

[15] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2-3):79–91, 2004.

[16] J. Peters, S. Vijayakumar, and S. Schaal. Reinforcement learning for humanoid robotics. In *Proc. of the Third IEEE-RAS Intl. Conf. on Humanoid Robots*, pages 1–20, 2003.

[17] M. Saggar, T. D'Silva, N. Kohl, and P. Stone. Autonomous learning of stable quadruped locomotion. *Lecture Notes in Computer Science*, 4434:98, 2007.

[18] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *Intl. Symposium on Robotics Research (ISRR2003)*. springer, 2004.

[19] R. Schmidt and T. Lee. *Motor Control And Learning: A Behavioral Emphasis*. Human Kinetics, 2005.

[20] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich. Common metrics for human-robot interaction. In *Proc. of the 1st ACM SIGCHI/SIGART Conf. on Human-robot interaction*, pages 33–40. ACM New York, NY, USA, 2006.

[21] A. Thomaz and C. Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6-7):716–737, 2008.

[22] R. Zhang and P. Vadakkepat. An evolutionary algorithm for trajectory based gait generation of biped robot. In *Proc. of the Intl. Conf. on CIRAS*, 2003.