2013 13th IEEE-RAS International Conference on
Humanoid Robots (Humanoids).
October 15 - 17, 2013. Atlanta, GA

Iterative Learning Control for High-Fidelity Tracking of Fast Motions on Entertainment Humanoid Robots

Pranav A. Bhounsule and Katsu Yamane

Abstract— **Creating animations for entertainment humanoid robots is a time consuming process because of high aesthetic quality requirements as well as poor tracking performance due to small actuators used in order to realize human size. Once deployed, such robots are also expected to work for years with minimum downtime for maintenance. In this paper, we demonstrate a successful application of an iterative learning control algorithm to automate the process of fine tuning choreographed human-speed motions on a 37 degree-of-freedom humanoid robot. By using good initial feed-forward commands generated by experimentally-identified joint models, the learning algorithm converges in about 9 iterations and achieves almost the same fidelity as the manually fine tuned motion.**

I. Introduction

Humanoid robots for entertainment are designed to imitate human or animal movements. Their motions are usually choreographed by hand to move in synchrony with pre-recorded speech or music. Animating such robots is a time consuming process. For instance, it took several weeks for an animator to finish the 255-second long animation for the 37-joint humanoid robot used in this work. Furthermore, once the robots are deployed, their animations have to be periodically re-examined and fine-tuned with minimum downtime to adjust for robot wear and part replacements.

In this paper, we develop a controller to enable high fidelity tracking of a given choreographed motion. More specifically, we apply a technique called Iterative Learning Control (ILC) to improve the tracking performance of the 37 degree-of-freedom (DOF) hydraulic robot shown in Fig. 1. More details about the robot hardware and software follow in Section III. Some of the key challenges associated with motion tracking on this humanoid are; 1) at the user level, we can only send position commands at 120 Hz; 2) the system has 37 DOF; 3) the algorithm should adapt to changes in the hardware properties due to wear and part replacements; and 4) the algorithm should be able to handle fast human-like motions.

Based on the hardware limitations and problem requirements, we decided to use Iterative Learning Control (ILC) [1]. For a given reference motion, ILC performs model-free refinement of feed-forward position commands by using the tracking errors at each trial. ILC is therefore well suited for applications involving repetitive disturbances and/or unmodeled dynamics. Section IV-A presents the details of the ILC algorithm used in this paper.

The authors are with Disney Research Pittsburgh, 4720 Forbes Avenue, Lower Level, Suite 110 Pittsburgh, PA 15213, U.S.A. {pab47,kyamane}@disneyresearch.com

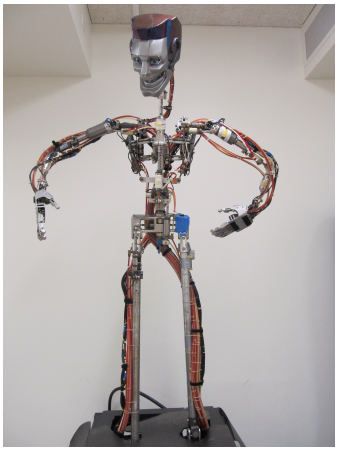

Fig. 1. The humanoid robot used as the experimental testbed for the research presented here.

An alternative would have been traditional controllers such as proportional, integral, and derivative control. While they can be applied to any reference motions, we assessed that it would be time consuming to find gains that would realize faithful tracking throughout the range of motion because of the limited sensing capability of the control system and the nonlinearity of the hydraulic actuators.

A key for achieving high performance trajectory tracking with ILC is obtaining good initial feed-forward commands. When a manually fine-tuned position command is not available, which is the case with animations originally created for films, we identify an input-output model of each joint. This model is then inverted to generate the initial feed-forward joint position command. Using ILC, we show that it is possible to approach the quality of a motion that has been manually fine tuned by a professional animator.

Figure 2 illustrates the issues with motion tracking on entertainment humanoid robots and how we tackle them in this paper. Each line on the plot represents either a position command, an actual joint trajectory, or a choreographed motion for one joint as a function of time. The choreographed motion is what the animator would like to realize on the robot.

In (a), the red solid line represents the manually fine-tuned joint position command that the animator obtains through trial-and-errors on the hardware in order to achieve the choreographed motion shown as the thick grey solid line. However, as the figure wears out or parts are replaced, the same joint position command produces a sluggish response

978-1-4799-2618-3/13/$31.00 ©2013 IEEE

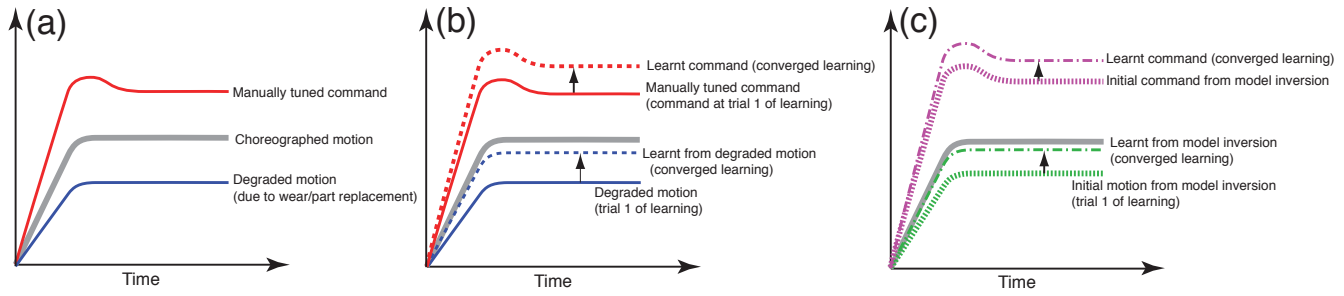

Fig. 2. Illustration of the conventional and proposed methods for animating entertainment humanoid robots. The plots show position commands, actual trajectories, and the choreographed (ideal) motion for one joint as a function of time. (a) An animator manually fine tunes the joint position command (red solid line) to achieve the choreographed motion (thick grey solid line). Due to wear or part replacement, the actual trajectory degrades over time (blue solid line). (b) We re-adjust the position command using ILC and obtain a new command (red dashed line) resulting in better tracking performance (blue dashed line). (c) If a manually tuned command is not available, we initialize the joint position command using the inverse of a joint model (magenta dotted line) and obtain the motion shown in green dotted line. We then use ILC to improve the tracking performance, resulting in the position command shown by the magenta dash-dotted line and joint motion shown by the green dash-dotted line. In our experience, (c) achieves better tracking performance with fewer iterations than (b).

(blue solid line).

Next, we illustrate how we use ILC to re-tune the motion. In (b), we start with the manually tuned joint position command (red solid line) and perform ILC using the error between the choreographed motion (thick grey solid line) and the degraded motion (blue solid line). As learning proceeds, we are able to drive the tracking error down leading to the converged joint motion (blue dashed line) and the converged joint position command (red dashed line).

In more practical situations, however, we may not have access to a manually tuned command, but instead only have the choreographed motion to realize. In this case, we use a model-based initialization as shown in (c), where we use the inverse of the joint model to generate the initial joint position command (magenta dotted line) which generates the motion shown by the green dotted line. ILC subsequently improves the tracking performance, converging at a new joint position command (magenta dash-dotted line) that gives better tracking result (green dash-dotted line). In our experience, model-based initialization followed by learning (c) converges faster and achieves better tracking performance compared to non-model-based initialization followed by learning (b).

II. Related Work

Generating fast motions on hydraulic robots is challenging especially because of the nonlinearities of the hydraulic dynamics. Boaventura et al. [2] used a detailed hydraulic dynamics model with pressure, force and position sensing to generate high performance locomotion of the IIT Hydraulic quadruped. Bentivegna et al. [3] studied position and force control of a single joint of a hydraulic humanoid robot and showed that high speed trajectory tracking is achievable using linear actuator model, rigid-body dynamics model, and full state (position, velocity and force) feedback. They also showed convincingly that simple proportional-derivative control on position gives a poor transient response such as large steady state errors at low gains and system instability at high gains. Since ILC uses tracking errors to repeatedly tune the feed-forward commands, one can use low gains while achieving good tracking performance and therefore avoid system stability issues [4].

So far, applications of ILC has been limited to learning fast motions of simple systems such as swing up of a inverted pendulums [5], [6], quadrotor flips [7], table-tennis paddle motions [8], pick and place operation of a two-axis manipulator [9], and planar motion tracking of 3–6 DOF industrial manipulators [10], [11], [12]. We believe that we are the first to apply ILC to successfully tracking fast motions of robots as complex as a 37-DOF humanoid robot.

III. System Description

A. Robot hardware

The humanoid robot we use in this research is shown in Fig. 1. The robot is on a fixed base, that is, its feet are clamped to the ground. The robot has 37 joints: 4 in the lower body (base bend, base sway, base rotate, and pelvis bend), 3 in the upper body (torso twist, torso bend, and torso sway), a linear degree of freedom in each shoulder, 7 in each arm (arm swing, arm rotate, arm bend, elbow bend, wrist swing, wrist rotate, and wrist bend), one in each of the 10 fingers, 3 in the head (head rotate, head nod, and head tilt) and 1 in the mouth. For actuation, the robot uses hydraulics operated at a pressure of 600 psi. Each joint has a rotary potentiometer or a linear variable differential transformer to sense joint position. Some of the valves have pressure transducers to measure the pressure difference, which roughly corresponds to the actuator force.

B. Controller architecture

A schematic diagram of the controller architecture is shown in Fig. 3. The controller consists of two levels: lower and higher levels. These levels differ in terms of sensor data, command inputs, and operating rates.

The lower level controller runs at a rate of 1KHz and senses the joint position and joint force. Each joint has its own lower-level controller. The joint velocity is obtained

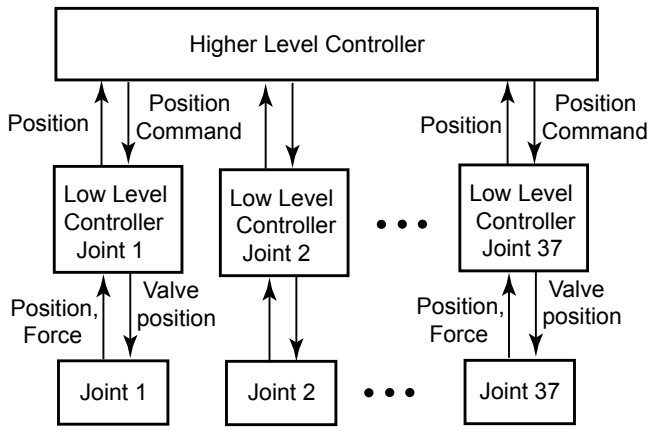

Fig. 3. Control software architecture of the experimental testbed.

by differentiation of the joint position followed by low-pass filtering of the differentiated joint position. The control command is the valve position v, which is computed as a linear function of joint position θ, joint velocity $\dot{\theta}$, and joint force F as follows:

$$v = -K_\theta(\theta - \theta_c) - K_{\dot{\theta}}\dot{\theta} - K_F F \tag{1}$$

where K_θ, $K_{\dot{\theta}}$, and K_F are the gains on the position, velocity, and force, and θ_c is the position command.

The higher level controller runs on an external PC and communicates with the lower-level controllers at a rate of 120 Hz. It can send position commands θ_c to the lower-level controllers, and read back the joint position and force measurements.

In order to emulate the in-field scenario, we assume that we cannot change the low-level controller and use the pre-installed default controller (1). Therefore, our only control variable is the position command θ_c for each joint.

IV. System Modeling and Control

A. Iterative learning control algorithm

Let i represent the trial number and j the time index that goes from 1 to n_j (end time). We denote feed-forward position command by $\theta_c^i(j)$ and the error between actual joint position and choreographed position by $e_c^i(j)$. Note that the feedforward command $\theta_c^i(j)$ and error $e_c^i(j)$ are vectors of joint angles and that learning is done on all joints simultaneously. The ILC algorithm we use is described is follows [13]:

1) For trial 1, set the error $e_c^1(j) = 0$ and initialize the feed-forward command $\theta_c^1(j)$ by either a given manually tuned command or the one obtained by inverting the system model described in Section IV-B.
2) For subsequent trials do:
   - Command execution: send the feed-forward commands $\theta_c^i(j)$ $(j = 1, 2, ..., n_j)$ to the robot and save the resulting tracking errors $e_c^i(j)$ $(j = 1, 2, ..., n_j)$.
   - Command modification: update the feed-forward command using the tracking error at trial i by $\theta_c^{i+1}(j) = \theta_c^i(j) + \gamma e_c^i(j)$, where γ is a manually tuned learning parameter.
3) Stop when an error metric $e_c^i(j)$ increase after an iteration. The learnt feed-forward command is then $\theta_c^{i+1}(j)$ $(j = 1, 2, ..., n_j)$.

An advantage of the ILC algorithm presented above is that the error update is done after the end of each trial. Hence it is possible to apply zero-phase filtering of errors [14] (using, for example, *filtfilt.m* in MATLAB) to remove high frequency noise from the error signal. Since the ILC algorithm is causal, we could have based our command modification on $e_c^i(j+1), e_c^i(j+2)$ and so on. However, we used only the $e_c^i(j)$ values to keep the ILC algorithm simple.

B. Modeling

The initial feed-forward command used in ILC can significantly influence the convergence rate of the algorithm and the final tracking performance as we will show in Section V. However, we may not always have access to good initial feed-forward command, especially if an animation is newly created and a lot of manual tuning is required to obtain such command. In this case, we use the inverse of the input-output model to initialize ILC.

We apply an input-output based black-box approach to fit a model to the individual joints. Our input signal is the position command θ_c and output is the actual joint position θ. We assume that the joint's dynamics and low-level controller can be described as a second-order model. We also know that the low-level controller has a communication time delay. The joint model is therefore given by

$$a_2\ddot{\theta}(t) + a_1\dot{\theta}(t) + a_0\theta(t) = \theta_c(t - T) \tag{2}$$

where T, a_2, a_1, a_0 are the parameters to be identified.

We identify the constants in the model (2) as follows. First, we set the individual joint approximately in the middle of its range of motion. Next, we excite the joint using white noise with a sufficiently large magnitude as θ_c. Finally, we use least square fit on the position data to determine the four constants.

We verify the quality of the identified parameters in two ways. First, we run the same identification process using chirp signals with frequencies of 0–2 Hz as input and find that the parameters are within 10% from those obtained using the white noise input. Second, we estimate the communication delay T independently using a step input and find it to be within 5% of that obtained using the white noise input.

There are a few important limitations in the identified model. If the gains of the lower level controller changes, the model parameters would also change because the identified model includes the response of the lower level controller. Also, our model ignores some terms of the rigid-body dynamics such as gravity, centripetal, and Coriolis forces.

To obtain the full-body model, we simply concatenate all the single joint models described by (2). We verify that

the joints are indeed decoupled as follows. We first track a reference motion at two adjacent joints individually using a feed-forward command signal obtained by the inverted model in (2). Next, we track the same motion at the two joints simultaneously using the same feed-forward commands. We find that the tracking errors in the latter case are about the same as those in the former, which implies that the two joints are indeed decoupled.

V. Results

A. Setup

The animation sequence used for the experiments is the first 30 seconds of a 4 min 15 sec long animation created and fine tuned by a professional animator. Our goal is to accurately reproduce the choreographed motion that the animator intended to play on the robot (the thick grey line in Fig. 2). However, we do not know the choreographed motion in this case because the fine-tuned animation we have gives the manually tuned command (the red solid line in Fig. 2(a)). We therefore assume that the animator was satisfied with the actual robot motion when the fine-tuned animation was given as position command.

We can then obtain the choreographed motion by measuring the joint positions while playing the fine-tuned animation. Because the measured motion contains noise that would not be present in real choreographed motion, we apply a low-pass Butterworth filter of a cutoff frequency of 2 Hz for forward and reverse passes to obtain noiseless, zero phase shift data. We also differentiate the raw position once to get the velocity and twice to get the acceleration using forward difference and apply the same zero phase filtering, which will be used later in generating the initial feed-forward command using the inverse of the model.

In our experiments, we consider two learning scenarios.

The first scenario concerns the case where the robot hardware properties have changed due to wear or part replacements. To emulate hardware degradation, we change the proportional gain of the lower level controller to a quarter of the original value for 26 of the 37 DOFs (excluding the 10 finger and mouth DOFs). Decreasing the proportional gains results in more sluggish motion compared to the choreographed motion. We then apply our learning algorithm to improve the feed-forward command and achieve better tracking performance.

The second scenario emulates the case where the robot has to reproduce an animation created entirely in software and therefore no manually-tuned command is available. To generate an initial feed-forward command for ILC, we input the position, velocity, and acceleration of the choreographed motion obtained above to the inverse of the joint model given in (2). The resulting output is used as the initial feed-forward command, which is subsequently refined using the ILC algorithm.

B. Experimental Results

We first conducted a set of preliminary experiments to determine the learning parameter γ. A small value of the learning parameter leads to slow learning that may not be practical. On the other hand, a large value leads to faster learning but can cause amplification of noise leading to system vibration and sometime may also drive the system unstable. We found that a large learning parameter lead to a distinct robotic-looking motion often seen in high gain position tracking in electrical robots. We manually tuned the learning gain ($\gamma = 0.5$) for one joint (left elbow bend) based on a trade off between noise amplification and algorithm convergence. We then used the same learning parameter for all the joints.

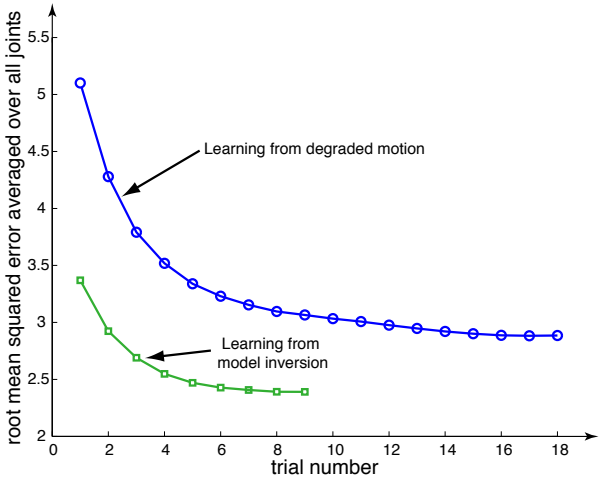

Fig. 4. Convergence of the ILC algorithm for learning from model based initialization (green squares) is twice faster and marginally better than learning from a non-model based initialization (blue circles).

Figure 4 shows the learning curve for the two learning scenarios. The error metric at iteration i is

$$e_i = \frac{1}{N}\sqrt{\frac{1}{n_j}\sum_{j=1}^{n_j} e_c^2(j)} \tag{3}$$

where $e_c(j)$ is the tracking error at time j and $N = 37$ is the number of DOF of robot. While learning from degraded motion takes 18 trails to converge, learning from model inversion takes only 9 trials and the final overall performance is marginally better than the other case. This result confirms that the model inversion gives better initial feed-forward command and that it results in faster convergence to better performance.

We also show the four robot motions in the accompanying video, snapshots (Fig. 5), and plots (Fig. 7) as follows:

1) Degraded motion: the first column in the video and Fig. 5, the blue solid lines in Fig. 7.
2) Learnt from degraded motion: the second column in the video and Fig. 5, the blue dashed lines in Fig. 7.
3) Learnt from model inversion: the third column in the video and Fig. 5, the green dash-dotted lines in Fig. 7.
4) Choreographed motion: the fourth column in the video and Fig. 5, the gray solid lines in Fig. 7.

The accompanying video show qualitative comparison between the four motions. Although it is difficult to discern

the difference when played at normal speeds, one can see differences when played at slow speeds. Delay and tracking error in the degraded motion are particularly visible, while the other three motions are visually similar. The delays and errors may be critical in synchronizing with music or pointing at an object in the environment.

Figure 5 shows some snapshots where we observed the motion between various hardware experiments to be distinctively different. In particular, observe the left hand positions and the head positions in the snapshots.

The plots in Fig. 7 give more quantitative comparison for selected joints. In the degradation scenario, the tracking error decreases by a factor of 2 for most joints. Furthermore, learning from model inversion generally results in even better tracking performance.

C. Discussion

In general, ILC can achieve tracking performance close to that of manually tuned commands. In some special cases, however, ILC still produces visually different motions. We point out two of such cases highlighted in Fig. 6 (not included in the video) and discuss possible solutions.

In the learnt motions at $t = 12$ s, we can observe that the head is placed differently from the choreographed motion. One might expect this to be due to a tracking error on the head turn degree of freedom, but this is not obvious in Fig. 7. The error is actually caused by aggregation of tracking errors at the torso and body turn degrees of freedom. In some cases, small tracking errors at the joint level can be magnified to cause large end-effector position and orientation errors. Our current controller cannot handle such situation because it operates independently on individual joints. A solution would be to perform ILC for the end-effector tracking errors, although it would require a kinematics model of the robot.

At $t = 16$ s, the left hand position is much higher in the manually fine tuned motion than in the other three motions. We notice that this error is due to the tracking error on left arm swing sideways as seen in Fig. 7. We speculate that this error has been caused by the gravity and may be further reduced by focusing on the error of this particular joint instead of considering the average error over all joints.

VI. Conclusion

In this paper, we demonstrated that a properly tuned Iterative Learning Control (ILC) algorithm can improve the tracking performance of high dimensional humanoid robots performing fast human-like motions under the presence of limited sensing and control bandwidth as well as nonlinear dynamics of hydraulic actuators. Furthermore, we confirmed that model-based approach to generating initial feed-forward command for ILC enables faster convergence and better tracking performance than non-model-based generation [15].

Our work would have the following impacts on animation, maintenance, and applicability of entertainment humanoid robots in general:

- Save animator time by offloading the task of fine tuning the motions on actual hardware. Note that animators would still need to choreograph the motions in computer software systems.
- Reduce downtime during maintenance by automatically adjusting for wear and part replacements.
- Allow smaller feedback gains in the low-level position controllers while achieving good tracking performance by feed-forward. Smaller gains make physical contacts easier because the robot will not exert excessive forces even when unexpected collisions occur, and therefore reduce maintenance issues.

Acknowledgment

Ross Stephens and Mouse Silverstein of Walt Disney Parks and Resorts helped with software bug fixes. David Feiten, a former employee of Walt Disney Imagineering, created the animation used in this work. Mike Mistry provided the interface routines to the low-level controller. We would also like to thank Chris Atkeson and Vinay Chawda for valuable suggestions and discussions.

References

[1] S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of robots by learning. *Journal of robotic systems*, 1(2):123–140, 1984.

[2] T. Boaventura, J. Buchli, C. Semini, M. Frigerio, and D. Caldwell. High performance hydraulic force control for articulated robots. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation*, 2013.

[3] D.C. Bentivegna, C.G. Atkeson, and J.Y. Kim. Compliant control of a hydraulic humanoid joint. In *IEEE-RAS International Conference on Humanoid Robots*, pp. 483–489, 2007.

[4] S. Daley, J. Hatonen, and D.H. Owens. Hydraulic servo system command shaping using iterative learning control. In *UKACC Control 2004 Mini Symposia*, pp. 117–121, 2004.

[5] A. Schöllig and R. D'Andrea. Optimization-based iterative learning control for trajectory tracking. In *Proceedings of the European control conference*, pp. 1505–1510, 2009.

[6] S. Jung and J.T. Wen. Nonlinear model predictive control for the swing-up of a rotary inverted pendulum. *Journal of Dynamic Systems, Measurement, and Control*, 126:666–673, 2004.

[7] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *2010 IEEE International Conference on Robotics and Automation*, pp. 1642–1648, 2010.

[8] M. Matsushima, T. Hashimoto, and F. Miyazaki. Learning to the robot table tennis task-ball control & rally with a human. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pp. 2962–2969, 2003.

[9] W. Messner, R. Horowitz, W-W. Kao, and M. Boals. A new adaptive learning rule. *IEEE Transactions on Automatic Control*, 36(2):188–197, 1991.

[10] M. Norrlof and S. Gunnarsson. Experimental comparison of some classical iterative learning control algorithms. *IEEE Transactions on Robotics and Automation*, 18(4):636–641, 2002.

[11] M. Norrlof. An adaptive iterative learning control algorithm with experiments on an industrial robot. *IEEE Transactions on Robotics and Automation*, 18(2):245–251, 2002.

[12] A. Tayebi and S. Islam. Adaptive iterative learning control for robot manipulators: Experimental results. *Control Engineering Practice*, 14(7):843–851, 2006.

[13] K.L. Moore. *Iterative Learning Control for Deterministic Systems, Advances in Industrial Control Series*. Springer Verlag, London, 1993.

[14] H. Elci, R.W. Longman, M. Phan, J-N. Juang, and R. Ugoletti. Discrete frequency based learning control for precision motion control. In *1994 IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pp. 2767–2773, 1994.

[15] C.G. Atkeson and J. McIntyre. Robot trajectory learning through practice. In *IEEE International Conference on Robotics and Automation*, volume 3, pp. 1737–1742, 1986.

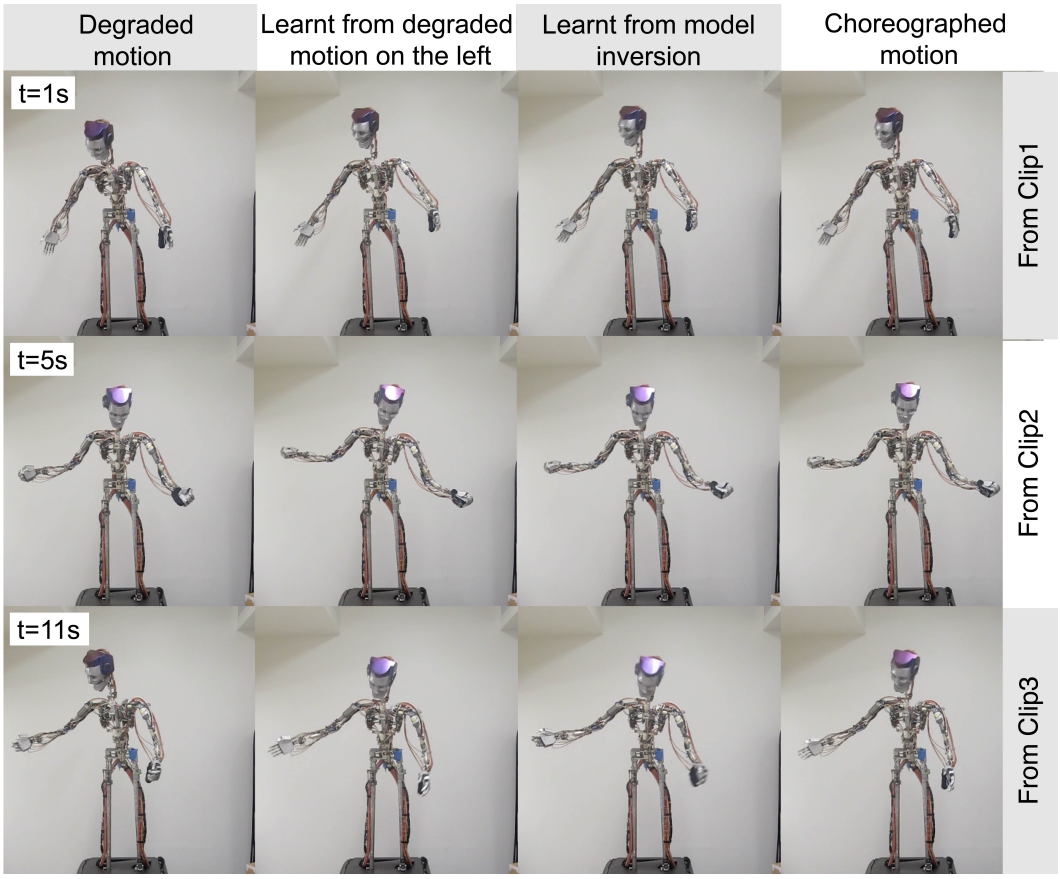

Fig. 5. Snapshots from the experiments using the four commands that highlight the advantage of learning. See submitted video for more information.

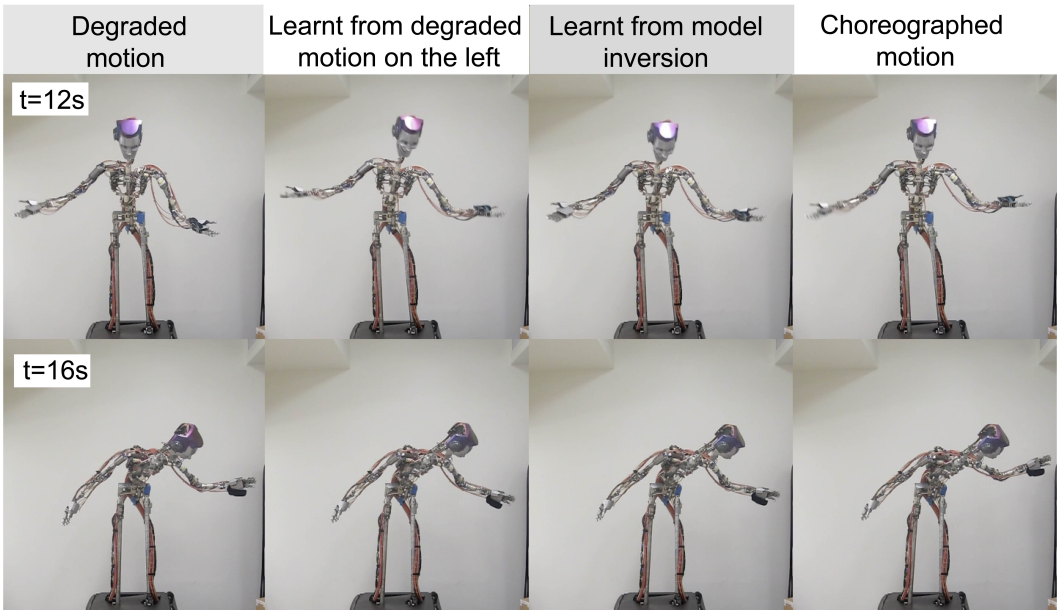

Fig. 6. Snapshots from the experiments using the four commands that highlight the limitations of learning (not included in the submitted video).

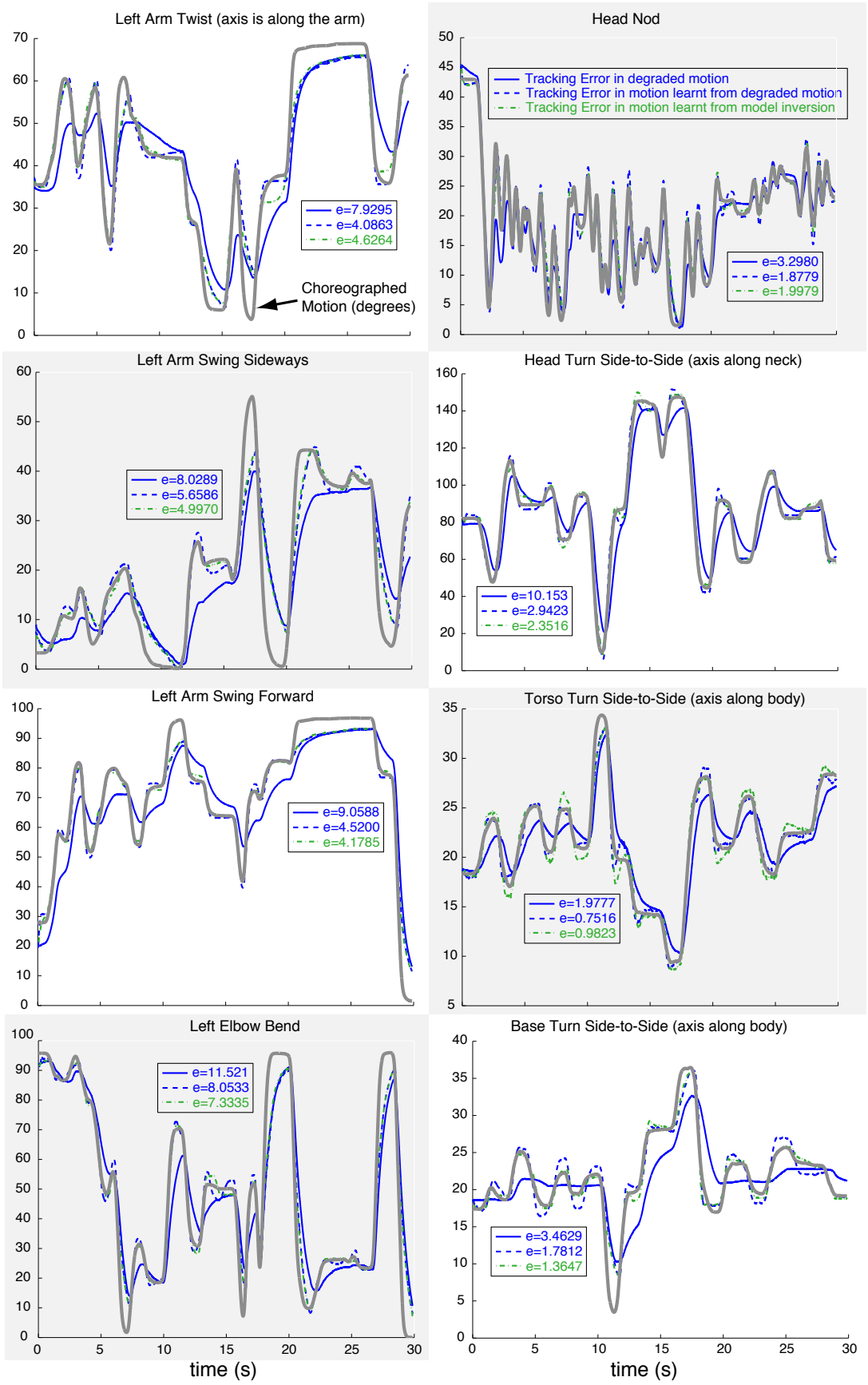

Fig. 7. Tracking performance for some of the joints