To appear in *Proceedings of the RoboCup International Symposium 2015* (RoboCup 2015), Hefei, China, July 2015.

A Study of Layered Learning Strategies Applied to Individual Behaviors in Robot Soccer

David L. Leottau[1], Javier Ruiz-del-Solar[1], Patrick MacAlpine[2], Peter Stone[2]

[1] Advanced Mining Technology Center, Department of Electrical Engineering, Universidad de Chile, Santiago, Chile
{dleottau,jruizd}@ing.uchile.cl
[2] Department of Computer Science, The University of Texas at Austin, Austin, TX 78712 USA
{patmac,pstone}@cs.utexas.edu

Abstract. Hierarchical task decomposition strategies allow robots and agents in general to address complex decision-making tasks. Layered learning is a hierarchical machine learning paradigm where a complex behavior is learned from a series of incrementally trained sub-tasks. This paper describes how layered learning can be applied to design individual behaviors in the context of soccer robotics. Three different layered learning strategies are implemented and analyzed using a ball-dribbling behavior as a case study. Performance indices for evaluating dribbling speed and ball-control are defined and measured. Experimental results validate the usefulness of the implemented layered learning strategies showing a trade-off between performance and learning speed.

Keywords: Reinforcement Learning, Layered Learning, Machine Learning, Soccer Robotics, Biped Robot, NAO, Behavior, Dribbling, Fuzzy Logic.

1 Introduction

The use of computational/machine learning (ML) techniques such as Reinforcement Learning (RL) allows robots, and agents in general, to address complex decision-making tasks. However, one of the main limitations of the use of learning approaches in real-world problems is the large number of learning trials required to learn complex behaviors. In addition, many times the learning of abilities associated with a given behavior cannot be directly used, i.e. combined or transferred to other behaviors. These drawbacks can be addressed by transfer learning [1] or hierarchical task decomposition strategies [2].

Layered Learning (LL) [3] is a hierarchical learning paradigm that enables learning complex behaviors by incrementally learning a series of sub-behaviors. LL considers bottom-up hierarchical learning, where low-level behaviors (those closer to the environmental inputs) are trained prior to high-level behaviors [4].

The main contribution of this paper is describing and analyzing how LL can be applied to design individual behaviors in the context of soccer robotics. Three different layered learning strategies are implemented and analyzed using the ball-dribbling behavior as a case study [5]. Ball-dribbling is a complex behavior where a robot play-

er attempts to maneuver the ball in a very controlled way while moving towards a desired target. Very few works have addressed ball dribbling behavior with humanoid biped robots [5–9]. Furthermore, few details are mentioned in these works concerning specific dribbling modeling [10, 11], performance evaluations for ball-control, or obtained accuracy to the desired path.

After modeling ball-dribbling behavior, some conditions needed to learn ball-dribbling under the LL paradigm are described. Afterwards, sequential, concurrent, and partial concurrent LL strategies are applied to the dribbling task and analyzed. Results from these experiments show a trade-off between performance and learning time, as well as between autonomous learning versus previous designer knowledge.

The paper is organized as follows: In Section 2 the Layered Learning paradigm and different LL strategies are detailed. Section 3 describes the ball-dribbling behavior, and Section 4 presents the application of the LL paradigm to the modeling and learning of ball-dribbling behavior. Experimental results are presented in Section 5, and conclusions are given in Section 6.

2 Layered Learning

Layered learning (LL) [3] is a hierarchical learning paradigm that enables learning complex behaviors by incrementally learning a series of sub-behaviors (each learned sub-behavior is a layer in the learning progression) [12]. LL considers bottom-up hierarchical learning, where high-level behaviors depend on behaviors in lower layers (those closer to the environmental inputs) for learning. From LL literature, three general strategies can be identified:

- **Sequential Layered Learning (SLL):** In the original formulation of the LL paradigm [3], layers are learned in a sequential bottom-up fashion. Lower layers are trained and then frozen (their behaviors are held constant) before advancing to learning of the next layer. While a higher layer is trained, lower layers are not allowed to change, which reduces the search space. However, it can also be restrictive because it limits the space of possible solutions that agents could search combining behaviors.
- **Concurrent Layered Learning (CLL):** CLL [4] allows lower layers to keep learning concurrently during the learning of subsequent layers. The agent may explore a behavior's joint search space combining all layers. Since CLL does not restrict the search space, its dimensionality increases, which can make the learning process more difficult.
- **Overlapping Layered Learning (OLL):** OLL [12] seeks to find a trade-off between freezing each layer once learning is complete (SLL) and leaving previously learned layers open (CLL). This extension of LL allows some, but not necessarily all, parts of newly learned layers to be kept open during the training of subsequent layers. In the context of learning parameterized behaviors this means that a subset of a learned behavior's parameters are left open and allowed to be modified during learning of the proceeding layer. The parts of previously learned layers left open "overlap" with the next layer being learned. Three general scenarios for overlap-

ping layered learning are distinguished in [12]: Combining Independently Learned Behaviors (CILB), Partial Concurrent Layered Learning (PCLL), and Previous Learned Layer Refinement (PLLR). This work considers the implementation of Partial Concurrent Layered Learning, where only part, but not all, of a previously learned layer's behavior parameters are left open when learning a subsequent layer with new parameters. The part of the previously learned layer's parameters left open is the "seam" or overlap between the layers [12].

3 Case Study: Soccer Dribbling Behavior

Soccer dribbling behavior with humanoid biped robot players is used as a case study [5]. Fig. 1 at left shows the RoboCup SPL soccer environment where the NAO humanoid robot [13] is used. The proposed modeling of dribbling behavior will use the following control actions: $[v_x, v_y, v_\theta]'$, the velocity vector; and the following state variables: ρ, the robot-ball distance; γ, the robot-ball angle; and, φ, the robot-ball-target complementary angle. These variables are shown in Fig. 1 at right, where the desired target (⊕) is located in the middle of the opponent's goal, and a robot's egocentric reference system is considered with the x axis pointing forwards. A more detailed description of the proposed modeling can be found in [5, 14].

Ball-dribbling behavior can be split into three sub-tasks which must be executed in parallel: *ball-turning*, which keeps the robot tracking the ball-angle ($\gamma = 0$), *target-aligning*, which keeps the robot aligned to the ball-target line ($\varphi = 0$); and *ball-pushing*, whose objective is that the robot walks as fast as possible and hits the ball in order to push the ball towards a desired target, but without losing possession of the ball. So, the proposed control actions are the requested speed to each axis of the biped walk engine, where $[v_x, v_y, v_\theta]'$ are respectively involved with *ball-pushing*, *target-aligning*, and *ball-turning* [15].

From a behavioral perspective, ball-dribbling can also be split in two more general tasks, *alignment* and *ball-pushing*. This division into two behaviors has been proposed in [5], based on the idea that *alignment* can be designed off-line, unlike *ball-pushing*, which needs interaction with its dynamic environment in order to learn a proper policy. In this way, *alignment* is composed of *ball-turning* and *target-aligning*. A behavior scheme of ball-dribbling is depicted in Fig. 2.(a).

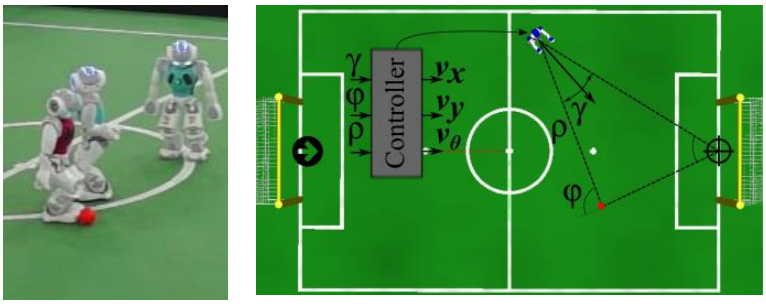

Fig. 1. A picture of the NAO robot dribbling during a RoboCup SPL game (left) and definition of variables for ball-dribbling modeling (right).

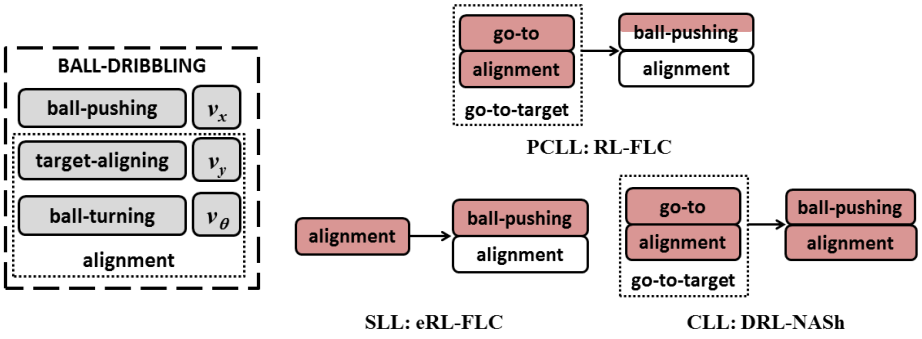

Fig. 2. (a) Behavioral scheme of the ball-dribbling problem. (b) Different layered learning strategies implemented; open behaviors are colored meanwhile frozen behaviors are white.

With respect to *ball-pushing*, the modeling of the robot's feet–ball–floor dynamics is complex and inaccurate because kicking the ball could generate several unexpected transitions, due to uncertainty of foot-ball interaction and speed when the robot kicks the ball (note that the robot's foot's shape is rounded and the foot's speed is different from the robot's speed v_x). Moreover, an omnidirectional biped walk intrinsically has a delayed response, which varies depending on the requested velocity $[v_x, v_y, v_\theta]'$. To learn when and how much the robot must slow down or accelerate is a complex problem, hardly solvable in an effective way with methods based on identification of system dynamics and/or kinematics and mathematical models [14]. To solve this problem as a Markov Decision Process (MDP) with an RL scheme for learning simultaneously, ball-dribbling dynamics have been successfully applied previously in the same domain [5, 14]. Thus, all the learning methods presented in this paper use an RL scheme for tackling the *ball-pushing* task.

4 Layered Learning of Dribbling Behavior

This section presents how three different strategies of the Layered Learning paradigm can be applied to the ball-dribbling task: PCLL, SLL, and CLL. These strategies are implemented by using a behavior in the first layer called *go-to-target*, where the robot goes to a desired target pose on the field. *Go-to-target* is composed in a very similar way to the ball-dribbling behavior depicted in Fig. 2.(a); it also uses *alignment* but uses *go-to* instead of *ball-pushing* as depicted at the top of Fig. 2.(b). *Go-to* behavior (see Table 1) is similar to *ball-pushing* as it also modifies v_x, but instead of directing the forward motion of the robot toward a ball it moves the robot forward toward a specific target location on the field. *Go-to-target* behavior is designed based on a Takagi-Sugeno-Kang Fuzzy Logic Controller (TSK-FLC) [16] which acts over the walk engine velocity vector. This behavior is currently part of the control architecture of the UChile Robotics Team [5, 17]. See Table 1 for descriptions of the behaviors' parameters and how they relate to each other.

Table 1. Summary of implemented behaviors and their learning methods

Behavior	LL strategy	What is learned in 1st layer	What is learned in 2nd layer
go-to	-	FLC parameters of v_x by using CMA-ES	-
alignment	-	FLC parameters of v_y and v_θ by using CMA-ES	-
go-to-target	-	*go-to* and *alignment*	-
Dribbling with RL-FLC	Partial concurrent (PCLL)	*go-to-target*	*ball-pushing*: A partial policy for v_x observing ρ, by using RL
Dribbling with eRL-FLC	Sequential (SLL)	*alignment*	*ball-pushing*: A policy for v_x observing $[\rho, \gamma, \varphi]'$, by using RL
Dribbling with DRL-NASh	Concurrent (CLL)	*go-to-target*	Three policies, for v_x, v_y, v_θ, which are learned in parallel observing the joint state $[\rho, \gamma, \varphi]'$, by using RL.
Dribbling with DRL	-	*ball-pushing* (v_x), *target-aligning* (v_y), and *ball-turning* (v_θ) by using Decentralized-RL [14]	-

For this work, the *go-to-target* controller parameters have been learned by using the RoboCup 3D simulation optimization framework of the LARG lab within the Computer Science Department at the University of Texas at Austin. This optimization framework uses the Covariance Matrix Adaptation Strategy (CMA-ES) [18], performed on a Condor [19] distributed computing cluster.

4.1 Partial Concurrent Layered Learning

The RL-FLC work reported in [5] proposes a methodology for modeling dribbling behavior by splitting it into two sub-problems: *alignment*, which is achieved by using a Fuzzy Logic Controller (FLC), and *ball-pushing*, which is learned by using a RL based controller. This methodology has been successfully used during RoboCup 2014 in the SPL robot soccer competitions by *UChile Robotics Team* [17] and it is currently the base of their dribbling engine.

Table 2. Description of states and actions for the RL-FLC scheme

States space: $s = [\rho]$				
		Min	*Max*	# bins
Feature	ρ	*0mm*	*600mm*	*13*
Actions space: $a = [v_x]$				
		Min	*Max*	# discrete actions
Action	v_x	*0mm/s*	*150mm/s*	*16*

The PCLL strategy is applied as follows: The *go-to-target* behavior is learned in the first layer for tuning FLC's parameters. During learning of the second behavior layer the entire *alignment* behavior is frozen while the *ball-pushing* behavior is partially re-learned. That means, only the parameter for how ρ affects v_x is opened to the RL agent, meanwhile parameters for how γ and φ influence v_x are kept frozen. So, γ and φ are not considered in the state space. Thus, ball-pushing parameters are partially refined in the context of the fixed *alignment* behavior. Please see top Fig. 2.(b) and Table 1.

Desired characteristics for a learned ball-dribbling policy are to have the robot walk fast while keeping the ball in its possession. That means ρ must be minimized (to keep possession of the ball), while at the same time maximizing v_x, which is the control action. Proposed RL modeling for learning the speed v_x depending on the observed state of ρ is detailed in Table 2. The proposed reward function is expressed in Eq.(1). This reward function reinforces walking forward at maximum speed ($v_{x.max\prime}$) without losing the ball possession ($\rho < \rho_{th}$).

$$r_x = \begin{cases} 1, & \rho < \rho_{th} \wedge v_x \geq v_{x.max\prime} \\ -1, & otherwise \end{cases} \quad (1)$$

4.2 Sequential Layered Learning

An enhanced version of the RL-FLC method is implemented using a SLL strategy. This enhanced approach (eRL-FLC) learns the *ball-pushing* behavior mapping the whole state space [ρ,γ,φ] by using a RL scheme. The modeling description is presented in [14]; it is designed to improve ball control because the former RL-FLC approach assumes the ideal case where target, ball, and robot are always aligned ignoring γ and φ angles, which is not the case during a real game situation.

The SLL strategy is applied as follows: The *alignment* behavior is learned in the first layer; then, during learning of the second layer, *alignment* is frozen and the whole *ball-pushing* behavior is learned by performing the ball-dribbling task in the context of the fixed *alignment* behavior. This is depicted at the bottom-left of Fig. 2.(b) and summarized in Table 1.

The proposed RL modeling is depicted in Table 3, where only *ball-pushing* is learned. The proposed reward function is expressed in Eq.(2).

Table 3. Description of States and Actions for eRL-FLC and DRL schemes

Joint state space: $s = [\rho, \gamma, \varphi]^T$				
		Min	*Max*	*# bins*
Feature$_1$	ρ	0mm	600mm	13
Feature$_2$	γ	-50°	50°	11
Feature$_3$	φ	-50°	50°	11
Actions space: $a = [v_x, v_y, v_\theta]$				
		Min	Max	# discrete actions
ball-pushing	v_x	0 mm/s	150 mm/s	21
target-aligning	v_y	-50 mm/s	50 mm/s	21
ball-turning	v_θ	-45 °/s	45 °/s	21

4.3 Concurrent Layered Learning

A Decentralized Reinforcement Learning (D-RL) strategy is proposed in [14], where each component of the omnidirectional biped walk $[v_x, v_y, v_\theta]'$ [20] is learned in parallel with single-agents working in a multi-agent task. Furthermore, this D-RL scheme is accelerated by using the Nearby Action Sharing (NASh) approach [15], which is introduced for transferring knowledge from continuous action spaces, when no information different to the suggested action in an observed state is available from the source of knowledge. In the early training episodes, NASh transfers actions suggested by the source of knowledge (former layer) but progressively explores its surroundings looking for better nearby actions for the next layer.

In order to learn dribbling behavior with the DRL-NASh approach, the CLL strategy is applied as follows: The *go-to-target* behavior is learned in the first layer. During learning of the second layer *go-to* and *alignment* behaviors parameters are left opened and relearned to generate *ball-pushing* and *alignment* behaviors, thereby transferring knowledge from *go-to-target* through use of the NASh method. This is depicted at the bottom-right of Fig. 2.(b) and summarized in Table 1.

Again, the expected policy is to walk fast towards the desired target while keeping the ball in the robot's possession. That means: maintaining $\rho < \rho_{th}$; minimizing $\gamma, \varphi, v_y, v_\theta$; and maximizing v_x. The proposed RL modeling is detailed in Table 3. The corresponding reward functions per agent are expressed in Eq.(2-4).

$$r_x = \begin{cases} 1, \rho < \rho_{th} \wedge |\gamma| < \gamma_{th} \wedge |\varphi| < \varphi_{th} \wedge v_x \geq v_{x.max'} \\ -1, \qquad otherwise \end{cases} \tag{2}$$

$$r_y = \begin{cases} 1, & |\gamma| < Ang_{th} \\ -1, & otherwise \end{cases} \tag{3}$$

$$r_\theta = \begin{cases} 1, & |\gamma| < Ang_{th} \wedge |\varphi| < Ang_{th} \\ -1, & otherwise \end{cases} \tag{4}$$

where $\rho_{th}, \gamma_{th}, \varphi_{th}$ are desired thresholds where the ball is considered controlled, meanwhile $v_{x.max'}$ reinforces walking forward at maximum speed.

5 Experimental Results and Analysis

5.1 Experimental Setup

As mentioned in the previous section, proposed LL schemes are implemented using the *go-to*-target behavior in the first layer, which is learned using CMA-ES. The second layer of all these schemes are performed by using a RL (SARSA(λ)) episodic procedure. After a reset, the robot is set in the center of its own goal (black right arrow in Fig. 1), the ball is placed in front of the robot, and the desired target is defined in the center of the opponent's goal ($\oplus$). The terminal state is reached if the robot loses the ball, or, the robot leaves the field, or, the robot crosses the goal line and reaches the target, which is the expected terminal state. Due to the comparative study purposes of this work, all the experiments are carried out in simulation. The training field is 6x4 meters. Ang_{th}=5°, $v_{x.max'} = 0.9 \cdot v_{x.max}$, and fault-state constraints are set as: $[\rho_{th}, \gamma_{th}, \varphi_{th}] = [500mm, 15°, 15°]$.

Four different learning schemes are presented in this paper: RL-FLC implemented with PCLL; eRL-FLC implemented with SLL; DRL-NASh implemented with CLL; and Decentralized RL scheme (DRL) as a base of comparison. The DRL scheme is proposed in [14] and briefly introduced in Table 1, it learns from scratch without any type of transfer learning or LL strategy.

The evolution of the learning process of each proposed scheme is evaluated by measuring and averaging ten runs. In this way, the following performance indices are considered to measure dribbling-speed and ball-control respectively:

- *% of maximum forward speed* ($\%S_{\text{Fmax}}$): given S_{Favg}, the average dribbling forward speed of the robot, and S_{Fmax}, the maximum forward speed: $\%S_{\text{Fmax}} = S_{\text{Favg}}/S_{\text{Fmax}}$.
- *% of time in fault-state (*$\%T_{\text{FS}}$*)*: the accumulated time in fault-state t_{FS} during the whole episode time t_{DP}. The fault-state is defined as the state when the robot loses possession of the ball, i.e., $\rho > \rho_{th} \vee |\gamma| > \gamma_{th} \vee |\varphi| > \varphi_{th}$, then: $\%\, T_{FS} = t_{FS}/t_{DP}$.
- *Global fitness (F)*: introduced for the sole purpose of evaluating and comparing both performance indices together. It is computed as follows: $F = 1/2 \cdot [(100 - \%S\text{Fmax}) + \%TFS]$, where *F=0* is the optimal policy.

5.2 Results and Analysis

Figure 3 shows the learning evolution of the four proposed schemes. Additionally, the policy of the run with the best performance from each scheme is tested and measured separately using 100 runs; average and standard error of those performances are presented in Table 4. The time to threshold index in Table 4 (learning speed) is calculated with a threshold of F=27%, according to global fitness plots in Fig. 3.

The time to threshold of the DRL scheme is the longest between all the tested schemes; this is the expected result, taking into account that no LL or transfer knowledge strategies have been implemented for this scheme. However, DRL learns

from scratch exploring the whole state-action space, allowing each sub-behavior (*ball-pushing*, *target-aligning*, and *ball-turning*) to learn about actions of the other two sub-behaviors. Even so, although DRL shows the lowest percentage of faults, it does not show the best global performance. The best performance is shown by the DRL-NASh scheme using CLL, which evidences the usefulness of CLL for this problem.

The DRL-NASh using CLL scheme shows the best global performance, the highest dribbling speed and the second best percentage of faults; however it takes on average around 1390 learning episodes before achieving asymptotic convergence, just around 13% faster than the DRL scheme. It validates the fact that by using concurrent layered learning it is possible to find better performance; the drawback is that increasing the search space dimensionality makes learning slower. Discussion about the NASh strategy and how the performance of first-layer-behavior influences the learning time and final performance is presented in [15]. Exploring this subject is a potential alternative to speed-up learning times when Concurrent LL is used with RL agents.

The RL-FLC using PCLL approach shows the fastest asymptotic convergence and the lowest accuracy. This is expected because RL-FLC is the least complex learning agent, which has frozen the major part of its search space, decreasing its performance but accelerating its learning.

Benefits of opening and learning the whole *ball-pushing* behavior for the eRL-FLC using SLL scheme are noticeable when observing standard deviation bars in Fig. 3. For this case, *ball-pushing* learns its policy interacting with *alignment* during the second layer of SLL, which does not dramatically increase the dribbling speed though it reduces the amount of faults, just as it was designed.

According to global fitness versus time to threshold in Table 4, a trade-off in terms of performance and learning speed can be noticed. Additionally, there is another non-measured but important trade-off between autonomous learning versus previous designer knowledge. Those LL strategies that reduce the search space's dimensionality require previous knowledge of the problem for determining effectively what part of former learned layers should be opened, and what type of LL strategy is better for each particular problem. On the other hand, more autonomous learning strategies as CLL or merely learning from scratch require less designer knowledge but can make learning difficult.

Some videos showing the learned policies for dribbling can be seen at[1]. Currently the learned policy is transferred directly to the physical robots, thus, the final performance is dependent on how realistic the simulation platform is. On the other hand, since state variables are updated and observed frame by frame acting like a closed loop control action, which tries to minimize the error, a different initialization of robot, ball, and target positions does not affect performance dramatically. The robot always tries to follow a straight-line between the ball and desired target emulating the training environment.

[1] https://www.youtube.com/watch?v=HP8pRh4ic8w
https://www.youtube.com/watch?v=_i8aNYSd6Iw&feature=youtu.be

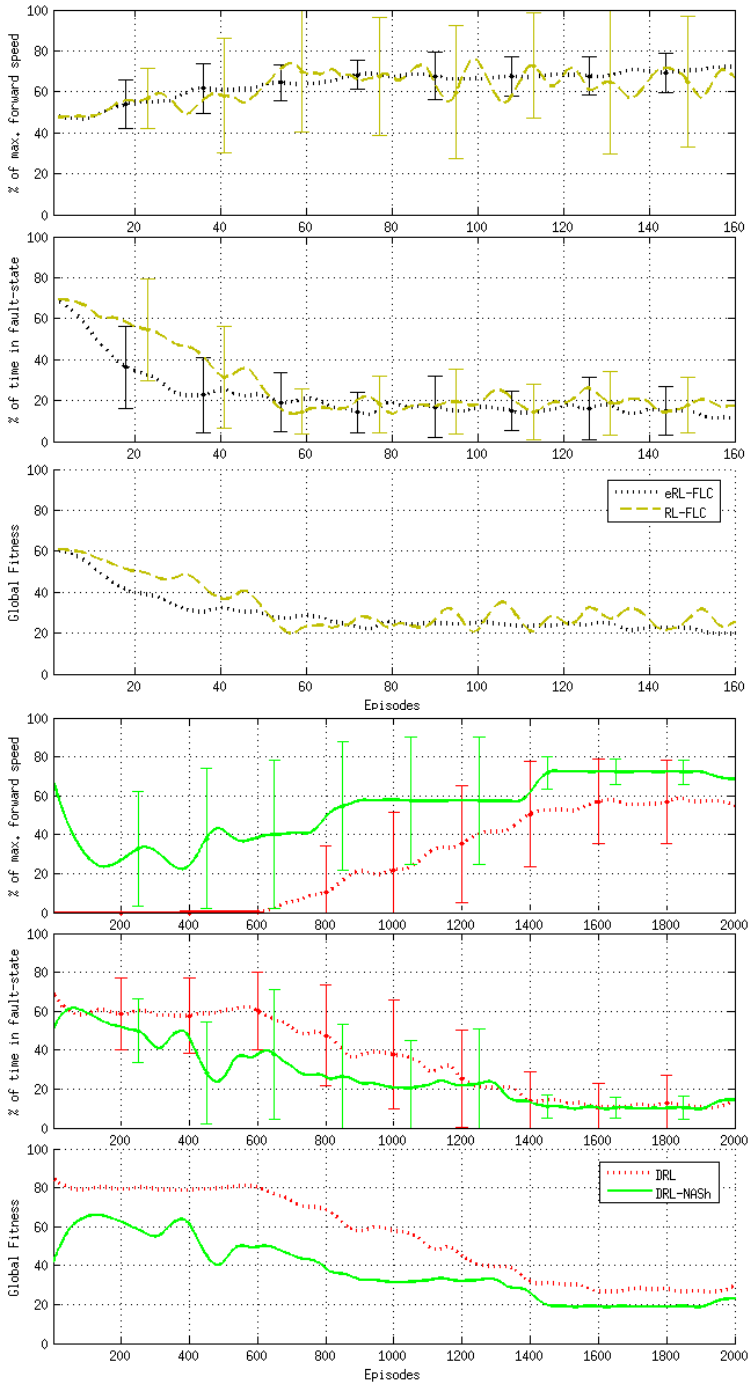

Fig. 3. Learning evolution with standard deviation bars of the four proposed schemes.

Table 4. Performance indices

Method	$\%S_{Fmax}$		$\%T_{FS}$		*F*	*Time to Th. (Episodes)*
	Avg.	*Std.Err*	*Avg.*	*Std.Err*	*Avg.*	
DRL-NASh (CLL)	74.83	0.049	14.69	0.080	19.92	1391
eRL-FLC (SLL)	61.49	0.032	16.84	0.061	27.67	66
RL-FLC (PCLL)	57.50	0.04	26.32	0.069	34.4	53
DRL	64.35	0.12	13.87	0.19	24.76	1594

6 Summary and future work

This paper has described how different Layered Learning strategies can be applied to design individual behaviors in the context of soccer robotics. Sequential LL, Partial Concurrent LL, and Concurrent LL strategies have been implemented and analyzed using ball-dribbling behavior as a case study.

Experiments have shown a trade-off between performance and learning speed. For instance, the PCLL scheme is capable of learning in around 53 episodes. This opens the door to make achievable future implementations for learning similar behaviors with physical robots. This is one of our short term goals and part of our future work.

Acknowledgments.
This work was partially funded by FONDECYT under Project Number 1130153. Research within the Learning Agents Research Group (LARG) at UT Austin is supported in part by NSF (CNS-1330072, CNS-1305287), ONR (21C184-01), and AFOSR (FA8750-14-1-0070, FA9550-14-1-0087). David Leonardo Leottau was funded under grant CONICYT-PCHA/Doctorado Nacional/2013-63130183.

References.

1. Taylor, M., Stone, P.: Transfer learning for reinforcement learning domains: A survey. J. Mach. Learn. Res. 10, 1633–1685 (2009).
2. Takahashi, Y., Asada, M.: Multi-controller fusion in multi-layered reinforcement learning. Multisensor Fusion and Integration for Intelligent Systems, 2001. MFI 2001. International Conference on. pp. 7–12 (2001).
3. Stone, P.: Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer. MIT Press (2000).
4. Whiteson, S., Stone, P.: Concurrent Layered Learning. Second International Joint Conference on Autonomous Agents and Multiagent Systems. pp. 193–200. {ACM} Press, New York, NY (2003).
5. Leottau, D.L., Celemin, C., Ruiz-del-solar, J.: Ball Dribbling for Humanoid Biped Robots: A Reinforcement Learning and Fuzzy Control Approach. In: Reinaldo A. C. Bianchi, H. Levent Akin, Subramanian Ramamoorthy, K.S. (ed.) RoboCup 2014: Robot World Cup XVIII - Lecture Notes in Computer Science 8992. pp. 549–561. Springer (2015).
6. MacAlpine, P., Barrett, S., Urieli, D., Vu, V., Stone, P.: Design and Optimization of an Omnidirectional Humanoid Walk: A Winning Approach at the RoboCup 2011 3D

Simulation Competition. Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12). , Toronto, Ontario, Canada, (2012).
7. Alcaraz, J., Herrero, D., Mart, H.: A Closed-Loop Dribbling Gait For The Standard Platform League. Workshop on Humanoid Soccer Robots of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids). , Bled, Slovenia (2011).
8. Meriçli, Ç., Veloso, M., Akin, H.: Task refinement for autonomous robots using complementary corrective human feedback. Int. J. Adv. Robot. Syst. 8, 68–79 (2011).
9. Latzke, T., Behnke, S., Bennewitz, M.: Imitative Reinforcement Learning for Soccer Playing Robots. In: Lakemeyer, G., Sklar, E., Sorrenti, D., and Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X SE - 5. pp. 47–58. Springer (2007).
10. Tilgner, R., Reinhardt, T., Kalbitz, T., Seering, S., Fritzsche, R., Eckermann, S., Müller, H., Engel, M., Wünsch, M., Mende, J., Freick, P., Stadler, L., Schließer, J., Hinerasky, H.: Nao-Team HTWK Team Description Paper 2013. RoboCup 2013: Robot Soccer World Cup XVII Preproceedings. RoboCup Federation, Eindhoven, The Netherlands (2013).
11. Röfer, T., Laue, T., Judith, M., Bartsch, M., Jenett, D., Kastner, T., Klose, V., Maaß, F., Maier, E., Meißner, P., Sch, D.: B-Human Team Description for RoboCup 2014. RoboCup 2014: Robot Soccer World Cup XVIII Preproceedings. , Joao Pessoa, Brazil. (2014).
12. MacAlpine, P., Depinet, M., Stone, P.: UT Austin Villa 2014: RoboCup 3D Simulation League Champion via Overlapping Layered Learning. AAAI-15 29th AAAI Conference on Artificial Intelligence. , Austin, Texas, USA (2015).
13. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: Mechatronic design of NAO humanoid. 2009 IEEE International Conference on Robotics and Automation. pp. 769–774. IEEE, Kobe, Japan (2009).
14. Leottau, D.L., Ruiz-del-solar, J.: An Accelerated Approach to Decentralized Reinforcement Learning of the Ball-Dribbling Behavior. AAAI-2015, Workshop on Knowledge, Skill, and Behavior Transfer in Autonomous Robots. pp. 23–29. , Austin, Texas USA (2015).
15. Leottau, D.L., Ruiz-del-Solar, J.: An Accelerated Approach to Decentralized Reinforcement Learning : A Humanoid Soccer Robots Validation. Intelligent Robots and Systems, 2015. IROS 2015. IEEE/RSJ International Conference on, submitted. , Hamburg, Germany (2015).
16. Takagi, T., Sugeno, M.: Fuzzy identi cation of systems and its application to modeling and control. IEEE Trans. Syst. Man, Cybern. 15, 116–132 (1985).
17. Yanez, J.M., Cano, P., Mattamala, M., Saavedra, P., Leottau, D.L., Celemin, C., Tsutsumi, Y., Miranda, P., Ruiz-del-solar, J.: UChile Robotics Team Team Description for RoboCup 2014. RoboCup 2014: Robot Soccer World Cup XVIII Preproceedings, July 2014. , Joao Pessoa, Brazil. (2014).
18. Hansen, N.: The CMA Evolution Strategy: A Tutorial, https://www.lri.fr/~hansen/cmatutorial.pdf.
19. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the Condor experience: Research Articles. Concurr. Comput. Pract. Exp. 17, 323–356 (2005).
20. Leottau, D.L., Yañez, J.M., Ruiz-del-solar, J.: Integration of the ROS Framework in Soccer Robotics: the NAO Case. In: Behnke, V., Veloso, M., Visser, A., and Xiong, R. (eds.) RoboCup 2013: Robot World Cup XVII, Lecture Notes in Computer Science Volume 8371. pp. 664–671. Springer (2014).