Ball Dribbling for Humanoid Biped Robots: A Reinforcement Learning and Fuzzy Control Approach

Leonardo Leottau, Carlos Celemin, Javier Ruiz-del-Solar

Advanced Mining Technology Center & Dept. of Elect. Eng., Universidad de Chile
{dleottau,carlos.celemin,jruizd}@ing.uchile.cl

Abstract. In the context of the humanoid robotics soccer, ball dribbling is a complex and challenging behavior that requires a proper interaction of the robot with the ball and the floor. We propose a methodology for modeling this behavior by splitting it in two sub problems: alignment and ball pushing. Alignment is achieved using a fuzzy controller in conjunction with an automatic foot selector. Ball-pushing is achieved using a reinforcement-learning based controller, which learns how to keep the robot near the ball, while controlling its speed when approaching and pushing the ball. Four different models for the reinforcement learning of the ball-pushing behavior are proposed and compared. The entire dribbling engine is tested using a 3D simulator and real NAO robots. Performance indices for evaluating the dribbling speed and ball-control are defined and measured. The obtained results validate the usefulness of the proposed methodology, showing asymptotic convergence in around fifty training episodes, and similar performance between simulated and real robots.

Keywords. Reinforcement Learning, TSK fuzzy controller, soccer robotics, biped robot, NAO, behavior, dribbling.

1 Introduction

In the context of soccer robotics, ball dribbling is a complex behavior where a robot player attempts to maneuver the ball in a very controlled way, while moving towards a desired target. In case of humanoid biped robots, the complexity of this task is very high, because it must take into account the physical interaction between the ball, the robot's feet, and the ground, which is highly dynamic, non-linear, and influenced by several sources of uncertainty.

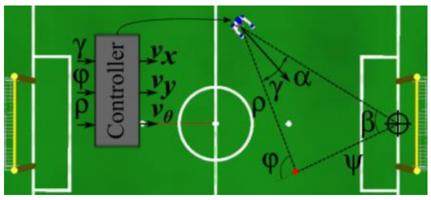
Very few works have addressed the dribbling behavior with biped humanoid robots; [1] presents an approach to incorporate the ball dribbling as part of a closed loop gait, combining a footstep and foot trajectory planners for integrating kicks in the walking engine. Since this work is more focused to the theoretical models and controllers of the gait, there is not included a dribbling engine final performance evaluation. On the other hand, [2] presents an approach that uses imitative reinforcement learning for dribbling the ball from different positions into the empty goal, meanwhile [3] proposes an approach that uses corrective human demonstration for augmenting a hand-coded ball dribbling task performed against stationary defender robots. Since these two works are not addressing explicitly the dribbling behavior, not many details about the specific dribbling modeling, or performance evaluations for the ball-control or accuracy to the desired target are mentioned. Some teams that compete in humanoid soccer leagues, such as [4], [5], have implemented successful dribbling behaviors, but to the best of our knowledge, no publications directly related about their dribbling´s methods have been reported for comparison. It is not clear whether in these cases a hand-coded or a learning-based approach has been used.

The dribbling problem has been addressed more extensively for the wheeled robots case, approaches based on the use of automatic control and Machine Learning (ML) has been proposed; [6], [7] apply Reinforcement Learning (RL), [8], [9] use neural networks and evolutionary computation, [10] applies a PD control with linearized kinematic models, [11] uses non-linear predictive control, and [12]–[14] apply heuristic methods. However, these approaches are not directly applicable to the biped humanoid case, due to its much higher complexity.

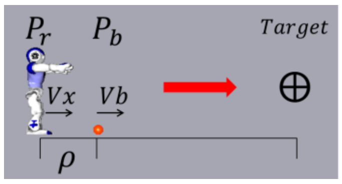
Although several strategies can be used to tackle the dribbling problem, we classify these in three main groups: (i) based on human experience and/or hand-code [2], [3], (ii) based on identification of the system dynamics and/or kinematics and mathematical models [1], [10], [11], and (iii) based on the on-line learning of the system dynamics [6]–[9]. In order to develop the dribbling behavior, each of these alternatives has advantages and disadvantages: (i) is initially faster to implement but vulnerable to errors and difficult to debug and re-tune when parameters change or while the system complexity increases; (ii) could be solved completely off-line by analytical or heuristic methods since robot and ball kinematics are known, but to identify the interaction between the robot's foot while it is walking, with a dynamic ball and the floor, could be anfractuous; in this way those strategies from (iii) which are capable to learn about that robot-ball-floor interaction, while find an optimal policy for the ball-pushing behavior, as RL, is a promise and attractive approach.

The main goal of this paper is to propose a methodology to learn the ball-dribbling behavior in biped humanoid robots, reducing as many as possible the on-line training time. In this way, the aforementioned alternatives (ii) are considered for reducing the complexity of behaviors learned with (iii). The proposed methodology models the ball-dribbling problem by splitting it in two sub problems, *alignment* and *ball-pushing*. The *alignment* problem consists of controlling the pose of the robot in order to obtain a proper alignment with the final ball's target. The *ball-pushing* problem consist of controlling the robot's speed in order to obtain, at the same time, a high speed of the ball but a low relative distance between the ball and the robot, that means controllability and efficiency. These ideas are implemented by three modules: (i) a fuzzy logic controller (FLC) for aligning the robot when approaching the ball (off-line designed), (ii) a foot-selector, and (iii) a reinforcement-learning (RL) based controller for controlling the robot's speed when approaching and pushing the ball (on-line learned).

Performance indices for evaluating the dribbling's speed and ball-control are measured. In the experiments the training is performed using a 3D simulator, but the validation is done using real NAO robots. The article is organized as follows: Section 2 describes the proposed methodology. Section 3 presents the experimental setup and obtained results. Finally, conclusions and future work are drawn in Section 4.



(a)



(b)

Fig. 1. Variables definition for: (a) the full dribbling modeling, (b) the *ball-pushing* behavior reduced to a 1-Dimensional problem.

2 A methodology for learning the ball-dribbling behavior

2.1 Proposed modeling

As mentioned in the former section, the proposed methodology splits the dribbling problem in two different behaviors: *alignment* and *ball-pushing*. Under this modeling, *ball-pushing* is treated as a one dimensional (1D) problem due to the ball must be pushed over the ball-target line when the robot is aligned; the *alignment* behavior is responsible to enforce this assumption, correcting every time the robot desired direction of movement.

The description of the defined behaviors will use the following variables: v_x, v_y, v_θ, the robot's linear and angular speeds; α, the robot-target angle; γ, the robot-ball angle; ρ, the robot-ball distance; ψ, the ball-target distance; β, the robot-target-ball angle; and, φ, the robot-ball-target complementary angle. These variables are shown in Fig. 1.(a). where the desired target (⨁) is located in the middle of the opponent goal, and with x axis pointing always forwards, measured in a robot's centered reference system. The behaviors are described as follows:

i. Alignment: in order to maintain the 1D assumption, it is proposed to implement a FLC which keeps the robot aligned to the ball-target line ($\varphi = 0, \gamma = 0$) while approaching the ball. The control actions of this subsystem are applied all the time over v_θ and v_y, and partially applied over v_x, only when the constraints of the 1D assumption are not fulfilled. Also, this behavior uses the foot selector for setting the foot that pushes the ball, in order to improve the ball's direction. Due to the nature of this sub-behavior, kinematics for the robot and ball can be modeled individually. Thus, we propose the off-line design and tuning of this task.

ii. Ball-pushing: following the 1D assumption, the objective is that the robot walks as fast as possible and hits the ball in order to change its speed, but without losing the ball possession. That means that the ball must be kept near the robot. 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 on the foot shape and speed when it kicks the ball (note that the foot's speed is different to the robot's speed v_x). Therefore it is proposed to model this behavior as a Markov Decision Process (MDP), in order to solve and to on-line learn it using a RL scheme. The behavior is applied only when the constraints of 1D assumption are fulfilled, i.e. when the robot's alignment is achieved. Fig. 1.(b) shows the variables used in this behavior.

Table 1. The v_x rule base.

$\gamma \setminus \varphi$	-H	-L	+L	+H
-H	$K_{Lx\rho}$	$K_{Lx\rho}$	$K_{Lx\rho}$	$K_{Lx\rho}$
-L	$K_{Hx\rho}$	$K_{Hx\rho}$	$K_{Hx\rho}$	$K_{Lx\rho}$
+L	$K_{Lx\rho}$	$K_{Hx\rho}$	$K_{Hx\rho}$	$K_{Hx\rho}$
+H	$K_{Lx\rho}$	$K_{Lx\rho}$	$K_{Lx\rho}$	$K_{Lx\rho}$

2.2 Alignment Behavior

The *alignment* behavior is compound of two modules: (a) the FLC which sets the robot speeds for aligning it to the ball-target line; and (b) the foot selector which depending on the ball position and robot pose decides which foot must kick the ball.

a) Fuzzy controller.

The FLC is inspired in a linear controller that tries to reduce φ and γ angles for being aligned to ball and target, while reduces ρ distance for approaching to the ball:

$$[v_x \quad v_y \quad v_\theta]^T = [(k_{x\rho} \cdot \rho) \quad (k_{y\varphi} \cdot \varphi) \quad (k_{\theta\gamma} \cdot \gamma - k_{\theta\varphi} \cdot \varphi)]^T \tag{1}$$

In order to perform better control actions for different operation points, constant gains of the three linear controllers can be replaced by adaptive gains. Thus, three Takagi-Sugeno-Kang Fuzzy Logic Controllers (TSK-FLCs) are proposed, which maintain the same linear controller structure for their polynomial consequents.

This linear controller and its non-linear counterpart based on FLC are proposed and compared in [15], please refers to that work for details about the proposed FLC. Table 1 depicts the rule base for the v_x FLC. Its consequent has only the gain $k_{x\rho}$, however the antecedent has the angle γ and φ. The rules basically set a very low gain if the robot is very misaligned to ball ($|\gamma|>>0$); else the robot goes straight and fast.

The v_y FLC's rule base is described in Table 2.(a). Based on (1), the control action is proportional to φ. The FLC makes adaptive the gain for φ; e.g., $k_{y\varphi}$ tends to zero where the ball is away, avoiding lateral movements which speed-up the gait.

The v_θ FLC's rule base is described in Table 2.(b). The control action is proportional to $(\gamma - \varphi)$, the FLC adapts the gain $k_{\theta\varphi}$, setting it close to zero when ρ is low or high, in that cases the robot is aligned to the ball minimizing γ; but when ρ is medium, the robot tries to approach the ball aligned to the target minimizing φ.

The fuzzy sets' parameters of each TSK-FLC are tuned by using the Differential Evolution (DE) algorithm [16]. It searches for solutions that minimize the fitness function F expressed in (2), whose performance indices is the time (t_f) used by the robot for achieving the ball being aligned to the target.

$$F = \frac{1}{S}\sum_{i=1}^{S} \frac{t_{f_i}}{(\rho_i)}, \tag{2}$$

where S is the total number of trained scenes with different initial robot and ball positions, whereas ρ_i is the $i-th$ initial Robot-Ball distance. Please refers to [15].

b) Foot Selector.

Since the proposed FLC is designed to align the robot with the ball-target line without using footstep planning, it cannot control which foot (right or left) is going to kick the ball. The FLC is designed to get the center of the ball aligned with the

Table 2. (a) The v_y rule base. (b) The v_θ rule base.

v_y **Controller rules**	v_θ **Controller rules**
If φ is Low & ρ is Low, then $k_{y\varphi}$ is Low	If ρ is Low, then $k_{\theta\gamma}$ is High, $k_{\theta\varphi}$ is Low
If φ is Low & ρ is High, then $k_{y\varphi}$ is Zero	If ρ is Med., then $k_{\theta\gamma}$ is Low, $k_{\theta\varphi}$ is High
If φ is High & ρ is High, then $k_{y\varphi}$ is Zero	If ρ is High, then $k_{y\varphi}$ is High, $k_{\theta\varphi}$ is Low
If φ is High & ρ is Low, then $k_{y\varphi}$ is High	

midpoint of the robot's footprints. This could generate undesired ball trajectories, because the NAO robot's foot-shape is rounded (see Fig. 2.b). This could be improved if the ball is hit with the front side of the foot, therefore the 1D assumption is more enforceable. Thus, it is proposed to align the robot to a point beside to the ball with an offset, a new biased position, called virtual ball (V). So, it is required to include a module that computes V and modifies the input variables depicted in the Fig. 1.(a).

Fig. 2 depicts the required variables for computing the position of the virtual ball, where $\oplus = [x_T \; y_T]', B = [x_b \; y_b]'$ are target and ball positions referenced to the local coordinates system of the robot, σ is the angle of the target-ball vector, and Ω is a unit vector with angle σ whis is calculated as:

$$\Omega = \begin{bmatrix} cos(\sigma) \\ sin(\sigma) \end{bmatrix} = \begin{bmatrix} x_b - x_T \\ y_b - y_T \end{bmatrix} \|B - \oplus\|^{-1} \qquad (3)$$

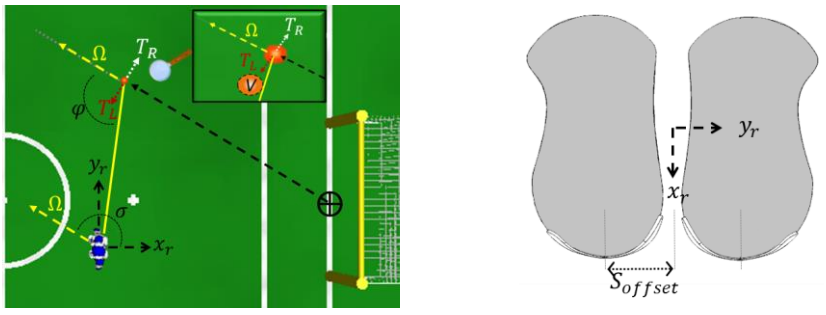
Due to the 1D assumption that robot achieves the ball aligned to the target (i.e. $\varphi = 0° \;\; \sigma = 180°$), depending of the selected foot for pushing the ball, robot should be displaced over its y axis towards left if the right foot is selected and vice versa. This sideward shifting is applied with orthogonal direction regarding Ω direction, it means to direction of translation vector T_L or T_R if left or right foot has been selected respectively. These vectors are shown in Fig. 2.(a). The 90º added to σ are positive for selecting left foot and negative in other case. The sideward shifting ($S_{sideward}$) expressed in (4) has an amplitude (S_{offset}) that depends of the physical structure of robot, it is the distance from the middle point between feet to the flattest edge as is shown in Fig. 2.(b).

$$S_{sideward} = \begin{bmatrix} cos(\sigma + f[k]90°) \\ sin(\sigma + f[k]90°) \end{bmatrix} S_{offset} \qquad (4)$$

The proposed criteria for selecting the foot is based on where the robot comes from, particularly depends of the sign of φ, e.g. in the case of Fig. 2.(a), the robot would have to select left foot for walking a shorter path towards a pose aligned to ball and target. Therefore, the sign of the angle added is opposite to the sign of φ. Equation (5) describes the rule for selecting the foot, where $f[k] = 1$ indicates left foot selected and $f[k] = -1$ the right one. In this rule some constraints are proposed for avoiding undesirable changes of the selected foot related to noisy perceptions: φ_h is a hysteresis for those cases when the φ angle is oscillating around zero, then the foot selected is changed only for considerable magnitude changes of φ; ρ_{th} is a threshold that avoids foot changes when robot is closer to the ball.

$$f[k] = \begin{cases} -sgn(\varphi), \rho > \rho_{th} \; \& \; |\varphi| > \varphi_h \\ f[k-1], \qquad otherwise \end{cases} \qquad (5)$$

The position of the virtual ball regarding the robot reference system, is given as $V = B + S_{sideward}$.



(a) (b)

Fig. 2. a) Angles and Vectors taken into account for computing the Virtual Ball (*V*). b) Footprint of NAO robot.

2.3 Reinforcement Learning for ball-pushing behavior

Since no footstep planners or specific kicks are performed, the ball is propelled by the robot's feet while it is walking, and the distance travelled by the ball depends on the robot's feet speed just before to hit it. Moreover, since our variables to be controlled are the robot speed relative to its center of mass, and not directly the speed of the feet, the robot's feet–ball dynamics turn complex and inaccurate. In this way, the RL of the *ball-pushing* behavior is proposed.

Speed based Modeling (*M1*). Our expected policy is walking fast while keeping the ball possession. That means to minimize ρ, and at the same time to maximize v_x. So, a first modeling for learning the speed v_x depending on the observed state of ρ is detailed in Table 3. Only one feature composes the state space, ρ, which is discretized with intervals of *50mm* (approximately the diameter of the ball in the SPL[1]). On the other hand, the robot speed v_x composes the actions space, so the agent has to learn about its self foot' dynamics handling v_x.

Acceleration based Modeling (*M2*). In this case, it is proposed to use speed increments or decrements as the action space for avoiding unreachable changes of speed in a short time and keeping a more stable gait. In this way, the expected policy is to accelerate for reaching the ball faster and to decelerate for pushing the ball soft enough. This modeling considers two additional features regarding *M1*: dV_{br}, the difference between ball (v_b)and robot speed (v_x), in order to track the ball; and v_x, in order to avoid ambiguous observed states and learn about the robot walking engine capabilities. It is expected that agent will be able to learn more about the dynamics between its walking speed, v_b, and ρ, regarding the selected foot to hit the ball. Since v_x is already a feature, it is possible to reduce the number of actions by using just three acceleration levels in the action space. The modeling for learning the acceleration acc_x is detailed in Table 4.

Reward function 1 (*R1*). The proposed reward $r_1(s,a)$ introduced in (6) is a continuous function which punishes the agent every step along the training episode. Since ρ is a distance and v_x a speed, its resulting quotient can be seen as the predicted

[1] http://www.informatik.uni-bremen.de/spl/bin/view/Website/Downloads

Table 3. States and Actions description for *M1*

States space: $s = [\rho]$, a total of *11* states.				
		Min	Max	Discretization
Feature	ρ	*0mm*	*500mm*	*50mm*
Actions space: $a = [v_x]$, a total of *5* actions.				
		Min	Max	Discretization
Action	v_x	*20mm/s*	*100mm/s*	*20mm/s*
There are *55* state-action pairs.				

time to achieve the ball assuming constant speed. So, the agent is punished according this time; it would be more negative if ρ is high and/or v_x is very low (not desired), otherwise $r(s,a)$ tends to zero (good reward). If the training environment is episodic and defines a terminal state, the cumulative reward would be better (less negative) if the agent ends the episode as fast as possible, always being near the ball.

$$r_1(s,a) = -\rho/v_x \qquad (6)$$

Reward function 2 (*R2*). The proposed reward $r_2(s,a)$ introduced in (7) is an interval and parametric function that punishes the agent when it loses the ball possession or when it walks slowly, and rewards the agent when it walks fast without losing the ball. *R2* could be more intuitive and flexible than *R1* because it includes threshold parameters to define an acceptable interval for ρ defined by ρ_{max}, and a desired minimum speed defined by V_{th}. Moreover, it is possible to increase the punishment or the reward according a specific desired performance. For example, to increase the punishment when $\rho > \rho_{max}$ in order to prioritize the ball control over the speed.

$$r_2(s,a) = \begin{cases} -1 & , if\ \rho > \rho_{max} \\ -1 & , else\ if\ v_x < V_{th}) \\ +1 & , otherwise \end{cases} \qquad (7)$$

The SARSA(λ) algorithm. The implemented algorithm for the *ball-pushing* behavior is the tabular SARSA(λ) with the replacing traces modification [17]. Based on previous work and after several trials, the SARSA(λ) parameters have been chosen prioritizing fastest convergences. In this way, the following parameters are selected: learning rate *α=0.1,* discount factor *γ=0.99,* eligibility traces decay *λ=0.9,* and epsilon greedy *ε=0.2* with exponential decay along the trained episodes.

3 Results

3.1 The ball-pushing behavior

The goal of this experiment is the reinforcement learning of the *ball-pushing* behavior. Fig. 3 depicts the initial robot positions (Pr_i) and the initial ball positions (Pb_i) for experiment i. In this case, the initial positions are set for $i = 1$ in order to configure a training environment similar to Fig. 1.(b). The terminal state is fulfilled where the robot crosses the goal line, then the learning environment is reset and a new episode of learning is started.

Table 4. States and Actions description for *M2*

States space: $s = [\rho \quad dV_{br} \quad v_x]$, a total of *110* states.				
		Min	Max	Discretization
Feature	ρ	*0mm*	*500mm*	*50mm*
Feature	dV_{br}	*-100*	*100*	*Negative or positive*
Feature	v_x	*20mm/s*	*100mm/s*	*20mm/s*
Actions space: $a = [acc_x]$, a total of 3 actions.				
		Min	Max	Discretization
Action	acc_x	*-20mm/s²*	*20mm/ s²*	*Negative, zero, and positive*
There are *330* state-action pairs.				

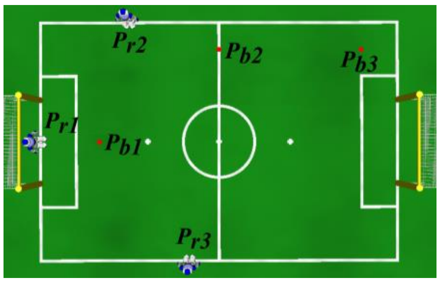


Fig. 3. Robot and ball initial positions for the tested Dribbling scenes. The field has 6x4m.

In section 2.3, two modeling and two reward functions have been proposed. The four possible combinations of them (M1-R1, M1-R2, M2-R1 and M2-R2) are used for learning the *ball-pushing* behavior. They are tested and compared by using the following performance indices:

- the episode time t_f, i.e. how long the agent takes to push the ball up to the target,
- the *% cumulated time of faults*: the cumulated time t_{faults} when the robot loses the ball possession, that means $\rho > \rho_{max}$, then:

$$\%\ cumulated\ time\ of\ faults = t_{faults}/t_f$$

- a global fitness function expressed as:

$$F = \frac{BTD+\psi_{tf}}{\psi_0}\int_{t=0}^{tf} \rho(t)/V_{rx}(t) \cdot dt\,,$$

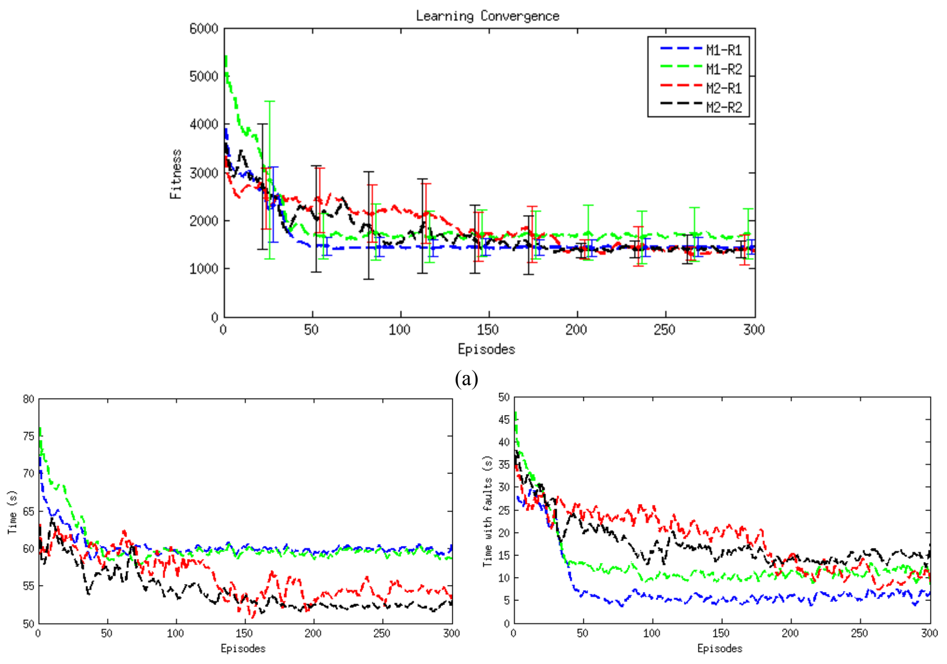
where *BTD* is the total distance traveled by the ball and ψ_{tf} is the ball-target distance when the episode is finished. In the ideal case $BTD = \psi_0$ and $\psi_{tf} = 0$.

The results of the experimental procedure for learning the *ball-pushing* behavior are presented in the Fig. 4. As it can be observed, there is a trade-off between convergence speed and performance. It is possible to see in Fig. 4.(a) those modeling using *M2* have the best performance while those modeling using *M1* achieve the fastest convergence. Also, from Fig. 4.(b) it can be noticed that *M2* prioritizes the dribbling speed meanwhile modeling with *R1* cares the ball possession as shown Fig. 4.(c). The best performance is achieved with *M2* and reward function *R1*, but its convergence is the worst. The fastest convergence is obtained with *M1*, independently from the reward function being used. Although when using *M2-R1* the final performance is almost 10% better than the obtained with *M1-R1*, Fig. 4.(a) shows a time reduction in learning convergence of 75% with *M1* respect to *M2-R1*. This means that *M1* is more convenient for learning with a physical robot.

Fig 4.(b) shows that *M1* carries out the dribbling about 15-20% slower than *M2*. However, *M1-R1* cares more for the ball possession (Fig. 4.(c)), which implies walking with lower speed and off course more time is taken. Notes that *M1-R1* gets about a half of the time of faults compared with the other three modeling as the Fig. 4.(c) depicts.

Modeling *M1* is simpler, less state-action pairs, so, it learns faster. Modeling *M2* has three features and more state-action pairs, so, it learns slower than *M1* but improves its performance. On the other hand reward *R1* is simpler, for both modeling obtains better performance, but, since it is less explicit for a detailed task, it convergence time is slower.

After to carry out several training episodes, testing different types of rewards and learning parameters for the proposed *ball-pushing* problem, we have concluded that the use of parameterized and interval rewards as *R2* is a very sensitive problem to small parameter changes. For example, a right selection of the magnitude of each interval reward, ρ_{max}, and V_{th} for handling the tradeoff between speed and ball-control, in addition to other learning parameters such as α, γ, λ and the exploration type, could dramatically modifies the learning performance.



(b) (c)

Fig. 4. Learning convergence through episodes by the modeling *M1* and *M2* using *R1* or *R2*, (a) global fitness average and intervals of confidence, (b) average of Time used for dribbling, (c) average of percentage of time with faults during dribbling.

Table 5. Validation results of the dribbling engine with three different scenes.

	Physical NAO			**Simulated NAO**	
	Dribbling Time (s)	**Time Increased (%)**	**Time Increased St.Dev.**	**Time Increased (%)**	**Time Increased St.Dev.**
Scene 1	53.71	38.56	11.48	31.91	3.62
Scene 2	49.57	49.57	11.79	37.88	3.31
Scene 3	44.38	32.39	10.06	9.98	3.07

3.2 Validation: the full dribbling behavior

For the final validation of the dribbling behavior, the policy learning with modelling *M1* and reward function *R1 (M1-R1)* has been selected because its best tradeoff between performance and convergence speed. Since the resulting policy is expressed as a Q-table with 55 state-action pairs, a linear interpolation is implemented in order to make a continuous input-output function. The FLC (*alignment*) and the RL based controller (*ball-pushing*) switches for handling v_x if the robot is or not into the *ball-pushing* zone. Finally, both controllers are transferred to the physical NAO robot, a hand parameter adjusts in the foot selector and FLC is carried out in order to compensate the so-called *reality gap* with the simulator.

The entire dribbling engine is validated using the 3D simulator SimRobot [4] and physical NAO robots, with the three different experiments/scenes described in Fig. 3, For each scene, the robot have to dribble the ball up to the target, a scene is finished when the ball cross the goal line. 50 runs are carried out in order to statistically significant results. For these tests, the performance indices are:

- the average of $t_{\mathrm{f},}$ the time that robot takes for finishing the dribbling scene,
- the average of *% time increased*, if t_{walk} is the time that robot takes to finish the path of the dribbling scene without dribbling the ball and walking at maximum speed, then: $\%\ time\ increased = (t_{\mathrm{f}} - t_{\mathrm{walk}})/t_{\mathrm{f}}$
- the standard deviation of the *% time increased.*

Table 5 shows the validation results of the entire dribbling engine with the simulator and the real robot. As it is usual, performances are better on simulation. This is more noticeable in scene 3, the most challenging for the motion and perception modules of the real robot. The dribbling time in scene 1 validates the asymptotic convergence values shown in figure 4.(b), in addition, for this particular case, the *% time increased* indicates that physical NAO spend 38% less time, if it walks without dribble the ball (i.e., ≈ 33s). The final performance of the designed and implemented dribbling engine can be watched in [18].

Figure 5 shows the learned policy for the robot speed (v_x). Assuming the minimum in v_x as the learned speed for pushing the ball ($v_{x\_push} \approx$ 40mm/s), it can be noticed that the agent learns to request v_{x_push} to the walking engine when ρ is biased from zero, $\rho_{biased} \approx$170mm. It can be interpreted as the agent learns about its own walking-request delays. Moreover, when the agent observes ρ less than ρ_{biased}, it learns to increase the speed, meanwhile the mentioned delay is over and the ball is pushed.

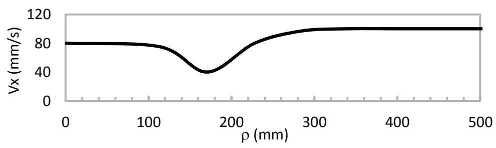


Fig. 5. Policy from modeling *M1; v_x* speed dependent on distance to the ball (ρ).

4 Conclusions and future work

This paper has presented a methodology for modeling the ball-dribbling problem in the context of humanoid soccer robotics, reducing as many as possible the on-line training time in order to make achievable futures implementations for learning the ball-dribbling behavior with physical robots.

The proposed approach is splitted in two sub problems: the *alignment* behavior, which has been carried out by using a TSK-FLC; and, the *ball-pushing* behavior, which has been learned by using a tabular SARSA(λ) scheme, a well-known, widely used and computationally inexpensive TD-RL method.

The ball-pushing learning results have shown asymptotic convergence in 50 to 150 training episodes depending on the state-action model used, which clarifies the feasibility of future implementations with physical robots. Unfortunately, according to the best of our knowledge, no previous similar dribbling engine implementations have been reported, in order to compare our final performance.

From the video [18], it can be noticed some inaccuracies with the alignment after pushing the ball. This could be related to the exclusion of the ball and target angles from the state space and reward function because the 1D assumption. Thus, as future work it is proposed to extend the methodology in order to learn the whole dribbling behavior avoiding switching between the RL and FLC. In this way, we plan to transfer the FLC policy of the pre-designed *alignment* behavior and refine it using RL in order to learn the *ball-pushing*. For that porpouses, transfer learning for RL is a promising approach. In addition, since the current state space is continuous and it will increase with the proposed improvements, RL methods with function approximation and actor critic will be considered.

Acknowledgments.
This work was partially funded by FONDECYT under Project Number 1130153 and the Doctoral program in Electrical Engineering at the *Universidad de Chile*.

References

1. Alcaraz, J., Herrero, D., Mart, H.: A Closed-Loop Dribbling Gait For The Standard Platform League. In: Workshop on Humanoid Soccer Robots of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids). Bled, Slovenia. (2011).
2. Latzke, T., Behnke, S., Bennewitz, M..: Imitative Reinforcement Learning for Soccer Playing Robots. In: Lakemeyer, G., Sklar, E., Sorrenti, D., Takahashi, T., eds. RoboCup 2006: Robot Soccer World Cup X SE - 5.Vol 4434. Lecture Notes in Computer Science. Springer Berlin Heidelberg. (2007)
3. Meriçli, Ç., Veloso, M., Akin, H.: Task refinement for autonomous robots using

complementary corrective human feedback. Int J Adv Robot Syst. 2011;8(2):68–79. (2013)

4. Röfer, T., Laue, T., Müller, J., et al.: B-Human Team Report and Code Release 2012. In: Chen X, Stone P, Sucar LE, der Zant T Van, eds. RoboCup-2012: Robot Soccer World Cup {XVI}. Springer Verlag ,Berlin, Heidelberg. (2012)
5. HTWK-NAO-Team: Team Description Paper 2013. In: RoboCup 2013: Robot Soccer World Cup XVII Preproceedings. Eindhoven, RoboCup Federation, The Netherlands. (2013)
6. Carvalho, A., Oliveira, R.. Reinforcement learning for the soccer dribbling task. In: Computational Intelligence and Games (CIG), 2011 IEEE Conference on. Seoul, Korea. (2011)
7. Riedmiller, M., Hafner, R., Lange, S., Lauer, M.: Learning to dribble on a real robot by success and failure. In: Robotics and Automation (ICRA), 2008 IEEE International Conference on. IEEE, Pasadena, California. (2008)
8. Ciesielski, V., Lai SYSY. Developing a dribble-and-score behaviour for robot soccer using neuro evolution. In: Work. Intell. Evol. Syst.; 2001:70–78. (2013)
9. Nakashima, T., Ishibuchi, H.. Mimicking Dribble Trajectories by Neural Networks for RoboCup Soccer Simulation. In: Intelligent Control, 2007. ISIC 2007. IEEE 22nd International Symposium on. (2007)
10. Li, X., Wang, M., Zell, A.: Dribbling Control of Omnidirectional Soccer Robots. In: Proceedings 2007 IEEE International Conference on Robotics and Automation. (2007)
11. Zell, A.: Nonlinear predictive control of an omnidirectional robot dribbling a rolling ball. 2008 IEEE Int Conf Robot Autom. (2008)
12. Emery, R., Balch, T.: Behavior-based control of a non-holonomic robot in pushing tasks. In: Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164).Vol 3. (2001)
13. Damas, B.B.D., Lima, P.U.P., Custodio, L.L.M.: A Modified Potential Fields Method for Robot Navigation Applied to Dribbling in Robotic Soccer. In: Kaminka, G., Lima, P., Rojas, R., eds. Rob. 2002 Robot Soccer World Cup VI SE - 6.Vol 2752. Lecture Notes in Computer Science. Springer Berlin Heidelberg. (2003)
14. Tang, L., Liu, Y., Qiu, Y., Gu, G., Feng, X.: The strategy of dribbling based on artificial potential field. In: 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE).Vol 2. (2010)
15. Celemin, C. Leottau, L.: Learning to dribble the ball in humanoid robotics soccer. Available at: https://drive.google.com/folderview?id=0B9cesO4NvjiqdUpWaWFyLVQ3anM&usp=sharing. (2014)
16. Storn, R., Price, K.: Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. (1995)
17. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press; (1998)
18. Leottau, L., Celemin, C.: UCH-Dribbling-Videos. Available at: https://www.youtube.com/watch?v=HP8pRh4ic8w. (Accessed: 28-Apr-2014)