INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 12(3), 330-346, June 2017.

Walking Motion Generation and Neuro-Fuzzy Control with Push Recovery for Humanoid Robot

P.E. Mendez-Monroy

Paul Erick Mendez-Monroy
IIMAS - Universidad Nacional Autonoma de Mexico
Parque CientĂfico y Tecnologico de Yucatan
Km. 5.5 Carretera Sierra Papacal - Chuburna
C.P. 97302 Sierra Papacal, Yucatan
erick.mendez@iimas.unam.mx

Abstract: Push recovery is an essential requirement for a humanoid robot with the objective of safely performing tasks within a real dynamic environment. In this environment, the robot is susceptible to external disturbance that in some cases is inevitable, requiring push recovery strategies to avoid possible falls, damage in humans and the environment. In this paper, a novel push recovery approach to counteract disturbance from any direction and any walking phase is developed. It presents a pattern generator with the ability to be modified according to the push recovery strategy. The result is a humanoid robot that can maintain its balance in the presence of strong disturbance taking into account its magnitude and determining the best push recovery strategy. Push recovery experiments with different disturbance directions have been performed using a 20 DOF Darwin-OP robot. The adaptability and low computational cost of the whole scheme allows is incorporation into an embedded system.
Keywords: push recovery, neuro-fuzzy systems, reinforcement learning, biped walking.

1 Introduction

With the recent growth of humanoid robots performing tasks within the human environment, research has focused on improving their movement and design for stable bipedal walking. Where dynamic walking is a general technique with predefined trajectories and feedback control to ensure that the robot is always rotating around a point in a base of support (BoS). To achieve dynamic walking the centre of gravity (COG) can be outside of the BoS of the robot, but the zero momentum point (ZMP) cannot. The ZMP is the point where the total angular momentum in the ankle is zero and the COG is the projection of the centre of mass (COM) on the ground, inside the BoS established by the feet. These concepts can be associated through an inverted pendulum model as the simplest model. But this technique is susceptible to external disturbance and irregularities in the surface, it requires strategies to maximise the balance and minimise the adverse effects of these disturbances.

The aim of push recovery is to perform an action that counteracts an external disturbance to prevent the robot from falling. The problem of push recovery has been researched using several approaches. Hyon *et al.* [5] use force-controlled actuators and a complete dynamic model to reject external disturbances. Park *et al.* [10] also use force-controlled actuators with optimisation algorithms for active balancing. Nevertheless, the requirements for multi-axis force sensors, an accurate model and intensive computing are their disadvantages.

Other research considers biomechanical movements using the knowledge of human behaviour in responses of unexpected disturbances with simple movements. [11], [17] [7]. Generally, these strategies are classified into three groups: Ankle, hip and stepping.

Some researchers have presented their work in push recovery when a robot walking in place. Pratt *et al.* [11] introduced the concept of "capture point" to determine where the foot should step after being pushed to return to its original pose. Pratt *et al.* [12] extended the concept to multiple steps.

The use of learning strategies to adapt the push recovery has increased in the last years. Rebula *et al.* [14] used capture point learning to improve simple step push recovery. While Missura *et al.* [8] proposed online learning to define the amplitude of the leg depending on the robot posture and the error in the desired step.

Some researchers have integrated several methods to complement the simple strategies for this level of complexity. Stephens [17] combines hip and ankle strategies with the purpose of balance control by defining a larger area return of a push and he established theoretical analytic bounds for both strategies. Hyon *et al.* [5] presented a multi-level posture balance method for humanoid robots and demonstrated push recovery with a biped SARCOS pushed from behind. Semwal *et al.* [15] used a hierarchical fuzzy control to decide by several push recovery strategies and used a simulated robot with good performance and low-cost computation but without online learning obtaining the same boundaries for each push recovery strategy as an analytic method.

However, few researchers have presented push recovery strategies in the walking phase, in order to continue the walking before a push. Komura *et al.* [6] proposed a feedback controller for the biped walking and applied a hip strategy as push recovery in a simulated robot. Wieber *et al.* [19] used a predictive control model to minimise the jerk and ZMP error, this improved the robustness to disturbances in walking.

The objective of this work is to design a practical strategy according to external disturbances using a neuro-fuzzy system to define the best push recovery behaviour to counteract the disturbance. The novel push recovery approach can maintain the robot walking, counteracting pushes from any direction and in any phase with a certain magnitude of disturbance. This scheme has the advantages of using a simple model and a controller of low computational cost. Furthermore, the design is applicable to generic robots with position controlled actuators, inertial measurement unit and joint position sensors.

The remainder of the paper proceeds as follows. Section 2.1 reviews three biomechanical push recovery controllers and their implementation on position-controlled actuators. Section 2.1 explains the details of the neuro-fuzzy system, structure, design and estimation. Section 2.1 describes the reinforcement learning process with a modified Levengerg-Marquardt algorithm. Section 2.1 explains the walking pattern. Section 6 shows the experimental results using the neuro-fuzzy system with reinforcement learning. The paper is finished with some concluding remarks and future work.

2 Biomechanical push recovery

Biomechanical studies of human walking have shown that humans perform three different movement patterns as recovery strategies (ankle, hip, step) to reject a sudden external disturbance. The ankle strategy modulates torque at the ankle joint to keep humans upright, the hip strategy converts unpredicted linear momentum into angular motion at the torso, and the stepping strategy moves the swing foot in the direction of the disturbance. Recent work has focused on using such biomechanical strategies [12] [17]. The next review of these biomechanical strategies assumes simplified physical models of the robot and explains how they can be implemented on real humanoid robots with position controlled actuators.

The ankle strategy maintains the COG in the BoS by applying torque to the ankle joints. For position controlled actuators, the torque at the ankle can be modified either by controlling the ZMP or modulating the target angle of the ankle servo. In this approach, the latter is used

as the strategy of push recovery in the ankle.

$$\delta_{zmp} = -K_a x_{a,swing} \tag{1}$$

where K_a is the ankle feedback gain and the $x_{a,swing}$ is the current position of the swinging ankle.

The hip strategy applies angular acceleration in the torso to generate a reaction force that returns the COG to the BoS from a elapse time $\{T_0, T_1\}$, decelerate in time $\{T_1, T_2\}$ and revert to the initial position slowly $\{T_2, T_3\}$, this control called "bang-bang" can be represented for position controlled actuators as

$$\delta\theta_t = \begin{cases} K_h & 0 \leq t < T_2 \\ K_h \frac{T_3 - t}{T_3 - T_2} & T_2 \leq t < T_3 \end{cases} \tag{2}$$

where $\delta\theta_t$ is the torso position bias, T_3 the time to complete the control, and K_h is the hip feedback constant.

For even larger disturbances, no amount of body torquing will result in push recovery. In this case, it is necessary to take a step. The stepping strategy moves the BoS by taking a step. This strategy looks for the point on the ground where the robot can step to bring it to its original speed or even full stop [17]. The new step (walking phase) or modified step (stance phase) is calculated using the relative target foot position x_{step} [1].

$$x_{step} = \dot{\theta}_x \frac{\sqrt{z_c}}{g} = K_s \tag{3}$$

The definition of a real analytic boundary to apply each strategy is difficult without the exact knowledge of the current system state as well as the kinematic and mechanic restrictions. In addition, its design becomes impractical if some of the parameters, such as the weight or COG change. Instead, the problem is formulated through learning techniques and the parameters are obtained online using past experiences. Some learning strategies have been performed in other robots [3], [18], where the aim is to learn a policy that maps robot states and establish appropriate actions.

3 Neuro-fuzzy system for push recovery

The push recovery is one of the most complex tasks for humanoid robots and it requires an adaptive intelligent controller, The push recovery method is achieved using simulation, the manipulation of the real system and with expert knowledge on intuitive biped push recovery. Thus, a neuro-fuzzy system is proposed to select the appropriate control parameters for the three strategies mentioned above. The selection uses the current walking state and the torso position. The fuzzy controller is designed using system constraints (eg. maximum torque for the actuators, maximum angles for the joints, maximum step size, etc.), expert knowledge and is updated using new reinforcement training with a low computational cost.

Figure 1 shows the architecture of the neuro-fuzzy system with an "actor-critic" reinforcement learning. The strategy is divided in the sagittal and the coronal planes but, only the sagittal plane will be described as the design for the coronal plane is similar. The controller determines a proper action of push recovery in the presence of an external disturbance using the current states $\{\theta_t, \dot{\theta}_t\}$ and the internal reinforcement signal $\hat{\hat{r}}$ from the predictor.

The scheme separates the push recovery system from the motion generator to simplify the design. Thus, the motion generator performs its normal movements when there is no external

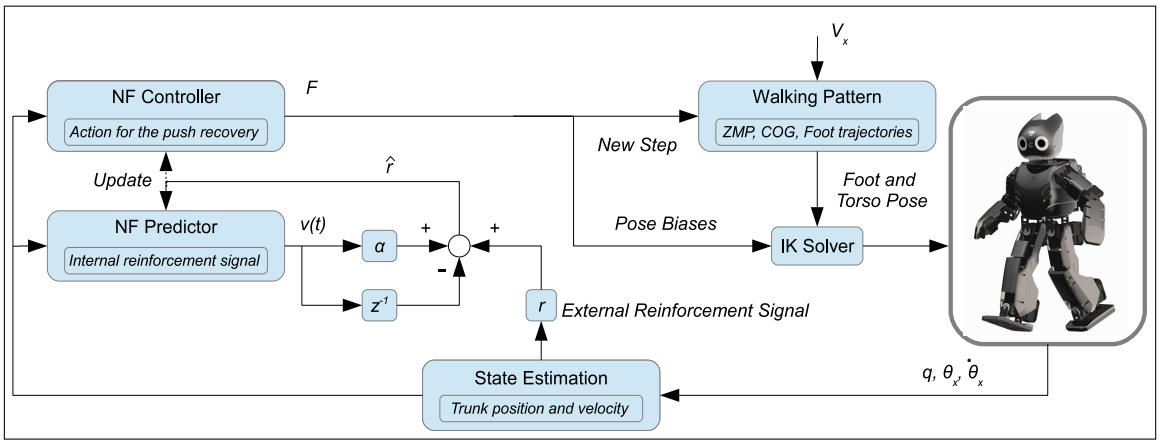

Figure 1: The neuro-fuzzy reinforcement structure

disturbances or unbalance. As soon as an external disturbance or unbalance occurs the neuro-fuzzy controller will establish the best learning strategy to counteract its effect.

The neuro-fuzzy controller uses the trunk position θ_t and its angular velocity $\dot{\theta}_t$ to define the feedback gains for the ankle, hip and stepping strategies. It is initialised offline using simulation, expert knowledge and the robot constraints (limit of the actuator torques, maximum step size, etc.) and it learns using reinforcement learning with low computational cost.

The neuro-fuzzy predictor has the same structure as the controller but with the internal reinforcement signal as the output. Its objective is to evaluate the performance of the whole neuro-fuzzy system using the external reinforcement signal (Reward) to update the parameters of the controller and predictor. Learning is performed by a modified Levenberg-Marquardt algorithm.

In the state estimation, the trunk position and angular velocity in the roll and pitch are estimated based on the accelerometer and gyroscope measurements and mixing with the measured joint angles (q) to use in the forward kinematic equations.

A walk pattern generator is presented to modify the walking trajectory according to the adopted push recovery strategy.

3.1 Neuro-fuzzy controller

The neuro-fuzzy controller (*"actor"*) can determine a better action for the next time instant based on the current system state and the internal reinforcement signal. The internal reinforcement signal from the predictor enables both the controller and the predictor to learn without waiting for external reinforcement, which may only be available at a time long after a sequence of actions has occurred. This scheme can speed up the learning of the complete system.

The fuzzy structure comprises five principal components: inputs, antecedent labels, rules, normalisation and consequent labels. Moreover, at the computational level, a fuzzy system can be seen as a feedforward neural network with five layers. Thus, The fuzzy structure is defined in five layers with node's basic functions for an online training.

Each layer has a input vector $\{u^k_{1\ldots p}\}$, a parameter vector $\{w^k_{1\ldots p}\}$ and a output vector $\{o^k_{1\ldots q}\}$ defining a activation function.

$$u^k_i = f\left(u^k_1, u^k_2, \cdots, u^k_p, w^k_1, w^k_2, \cdots, w^k_p\right) \tag{4}$$

$$o^k_j = a_j(u^k_i) \quad where\ a(\cdot)\ is\ the\ activation\ function \tag{5}$$

where $k = \{1, ..., 5\}$ is the kth layer, with p inputs and parameters for the jth output of the layer.

The next paragraph describes the functions of the nodes in each of the five layers of the proposed neuro-fuzzy controller with $X = \{x_1, ..., x_{in}\}$ inputs, $Y = \{y_1, ..., y_{out}\}$ outputs with r defined fuzzy rules like:

$$R_j : \textbf{if } x_1 \textit{ is } \mu_{1,j} \textit{ and ... and } x_{in} \textit{ is } \mu_{in,j} \textbf{ then } y_1 \textit{ is } \nu_{1,j} \textit{ and ... and } y_{out} \textit{ is } \nu_{out,j} \tag{6}$$

where μ_{ij} and ν_{lj} are fuzzy sets, the indexes $i = \{1, ..., in\}$ inputs, $l = \{1, ..., out\}$ outputs and $j = \{1, ..., r\}$ fuzzy rules.

Layer 1 (Inputs): The nodes in this layer transmit the inputs directly to the next layer (the vector parameters has unity values $w_i^1 = 1 \ \forall i$) eq.(7).

Layer 2 (Antecedents): Each single node to perform a membership function. The Gaussian membership function is used in this work eq.(8). where m_{ij}^2 and σ_{ij}^2 correspond to the centre and the width of the Gaussian function of the jth rule with the ith input variable.

Layer 3 (Rules): Implements the conjunction of all or some antecedent conditions in the jth rule. Hence, the rule nodes should perform the fuzzy "*and*" operation, in this case, the product function is used, in fuzzy theory, it is called *fire strength* eq.(9).

Layer 4 (Normalised): The number of nodes in this layer is equal to that of the rules layer. The effect of the layer is to perform a normalisation on each rule. It is called *normalised fire strength* for the jth rule eq.(10).

Layer 5 (Consequent): This layer will have as many nodes as there are output variables. Each output node combines the normalised fire strength with the fuzzy action and the defuzzification is computed with the centre of area method eq.(11) [4].

$$f_i^1 = u_i^1 \quad and \quad a_i^1 = f_i^1 \quad i = 1, 2, ..., in \tag{7}$$

$$f_{ij}^2 = \mu_{ij}\left(a_i^1;\ m_{ij}^2,\ c_{ij}^2\right) = -\frac{\left(a_i^1 - m_{ij}^2\right)^2}{\left(\sigma_{ij}^2\right)^2} \quad and \quad a_{ij}^2 = exp(f_{ij}^2) \tag{8}$$

$$f_j^3 = \prod_{i=1}^{2} a_{ij}^2 \quad and \quad a_j^3 = f_j^3 \quad w_{ij}^3 = 1 \ \forall i \tag{9}$$

$$f_j^4 = \frac{a_j^3}{\sum_{j=1}^{r} a_j^3} \quad and \quad a_j^4 = f_j^4 \quad where \quad \sum_{l=1}^{r} a_j^4 = 1 \tag{10}$$

$$f_l^5 = \frac{\sum_{j=1}^{r}(m_{lj}^5 \sigma_{lj}^5) a_j^4}{\sum_{j=1}^{r} \sigma_{lj}^5 a_j^4} \quad and \quad a_l^5 = f_l^5 \tag{11}$$

The complete fuzzy system F with fuzzy rules (7) is:

$$F = \{F_l,\ l = 1, ..., out\}, \quad F_l = \frac{\sum_{j=1}^{r} \nu_{lj}^{-} \prod_{i=1}^{in} \mu_{ij}(x_i)}{\sum_{j=1}^{r} \prod_{i=1}^{in} \mu_{ij}(x_i)}, \quad \nu_{lj}^{-} = \frac{\sum_{j=1}^{r}(m_{lj}^5 \sigma_{lj}^5) a_j^4}{\sum_{j=1}^{r} \sigma_{lj}^5 a_j^4} \tag{12}$$

where ν_{lj}^{-} is the centre of the lth output fuzzy set and the jth fuzzy rule. Thus, F is the fuzzy control action with the parameter vector $p_c = \{m_{ij}^2,\ \sigma_{ij}^2,\ m_{lj}^5,\ \sigma_{lj}^5\}$.

The parameter vector $p_c = \{m_{ij}^2,\ \sigma_{ij}^2,\ m_{lj}^5,\ \sigma_{lj}^5\}$ of the all weights in the controller is updated by minimising the internal reinforcement signal $\hat{r}$, such that, the system ends up with a good recovery strategy and avoids failure. The learning rule used to adjust the parameter vector is

a gradient-based adaption δp_c and Levenberg Marquardt with the objective function E_c to be minimised in the controller.

$$\delta p_c = \eta_c \frac{\partial E_c}{\partial p_c} = \eta_c \frac{\partial E_c}{\partial v} \frac{\partial v}{\partial F} \frac{\partial F}{\partial p_c}, \qquad E_c(t) = \frac{1}{2} v^2(t) \tag{13}$$

where η_c is the learning rate factor and v is the prediction of the external reinforcement signal.

3.2 Neuro-fuzzy predictor

To update a neuro-fuzzy system supervised learning [4] is the most common strategy used. These schemes need accurate training data to indicate the correct desired output to compute the training error. But, in the case of biped robots is difficult to get the correct output for each push recovery strategy because is a dynamic system with simplifications and uncertainties in the model and controller.

However, most of the real systems can provide a right-wrong binary decision (reinforcement signal) based on the current control, making a reinforcement training a better option to learn. Thus, a neuro-fuzzy predictor (*"critic"*) is used to determinate the parameter update using the external reinforcement signal r and to predict an internal reinforcement signal $\hat{r}$. The predictor structure is the same as the controller but only with the internal reinforcement signal like output.

The learning algorithm will be applied to the fuzzy controller and the fuzzy predictor simultaneously and only conducted by an external reinforcement feedback (R) using an orbital energy.

$$R = \frac{\dot{\theta}_x^2 + w^2 \theta_x^2}{2} \tag{14}$$

This is the "critic" information available for learning and indicates whether the output is right or wrong. In this paper, R is mapped to a range $r = \{-1 \rightarrow failure, 1 \rightarrow success\}$ to get a continuous reinforcement signal. Thus, the predictor output v approximates the discounted total reward-to-go $\tilde{R}(t) = r(t+1) + \alpha r(t+2)...$ [16]. It is the future accumulative reward-to-go, α is a discount factor for the infinite-horizon problem and $r(t+1)$ is the external reinforcement.

Defining the prediction error E_p and the objective function to be minimised as

$$e_p(t) = \alpha v(t) - v(t-1) - r(t) \qquad E_p(t) = \frac{1}{2} e_p^2(t) \tag{15}$$

Therefore, the parameter vector $p_p = \{m_{ij}^{2p},\ \sigma_{ij}^{2p},\ m_j^{5p},\ \sigma_j^{5p}\}$ is updated using a a Levenberg Marquardt algorithm and gradient-based update rule given by

$$\delta p_p = \eta_p \frac{\partial E_p}{\partial p_p} = \eta_p \frac{\partial E_p}{\partial e_c} \frac{\partial e_c}{\partial v} \frac{\partial v}{\partial p_p} \tag{16}$$

3.3 State estimation

State estimation uses the joint angles measured by the encoders, the inertial acceleration and angular velocity of the trunk measured by the IMU and a forward kinematic model to reconstruct the whole-body pose of the robot. With the reconstruction of the whore-body pose is obtained the current COM and which leg is the standing in.

By fusing the gyroscope and accelerometer data is estimated the trunk position. The trunk position $\hat{\theta}_t = (\theta_{tx}, \theta_{ty})$ is expressed using the inertial acceleration vector $a = (a_x, a_y) \in [-1, 1]$

measured by the accelerometer, first, a raw angle estimate is obtained. Then, it is mixed with the angular velocity in the n iteration of the control loop (eq. 19).

$$rl\dot{\theta}_{tx} = arcsin(a_x) \tag{17}$$

$$\dot{\theta}_{ty} = arcsin(a_y) \tag{18}$$

$$\theta_n = (1-\kappa)(\theta_{n-1} + \rho(\hat{\dot{\theta}}_n - b_n)) + \kappa\hat{\theta}_n \tag{19}$$

where $\hat{\dot{\theta}}_n$ is the raw angular velocity, κ is a blending factor, ρ is the control time, and b_n is the gyro drift. There are other more advanced options for mixing the gyroscope and accelerometer measurements [2].

On the other hand, the measurements of the joints (q) obtained through the encoders of each motor are applied to a direct kinetic algorithm to get the current pose. This kinematic model is rotated around the centre of the supporting foot such that the trunk position equals the roll and pitch angles $\theta_q = (\theta_{tx}, \theta_{ty})$ obtained in equation (19).

Thus, the difference between the trunk angles achieved by the IMU and the joint angles is used by the neuro-fuzzy controller.

$$\begin{aligned} \Delta\theta &= \theta_t - \theta_q \\ \dot{\theta} &= \hat{\dot{\theta}} \end{aligned} \tag{20}$$

4 Reinforcement Levenberg-Marquardt learning

As mentioned earlier, a neuro-fuzzy system is used in both the controller and the predictor where their parameters are updated using reinforcement learning. According to eq.(13) and eq.(15), it is possible use a modification of the Levenberg-Marquardt algorithm [13] to speed up the update to the neuro-fuzzy system. The parameters $p = \{p_c, p_p\}$ are updated using normalisation in both networks to confine the parameter values into some appropriate range (21).

$$p(t+1) = \frac{p(t) + \delta p(t)}{\|p(t) + \delta p(t)\|_1} \quad with \quad \delta p(t) = -(H(p))^{-1} J^T(p) e(p) \tag{21}$$

where approximation of the Hessian matrix H and the Jacobian matrix is calculated using the backpropagation algorithm.

$$H(p) = J^T(p)J(p) + \mu I \quad with \quad J = [\frac{\partial E}{\partial p_1}, \frac{\partial E}{\partial p_2}, ..., \frac{\partial E}{\partial p_N}] \tag{22}$$

where p_0 are the initial parameter values for the controller and predictor, ε_1, ε_2 and ε_3 are specific stop criteria in the LM algorithm, k_{max} is the maximum iteration number and ε_4 is the controller or predictor error prediction. E_{new} is the value of the objective function evaluated with the new parameters.

The Jacobian matrix in eq.(22) for the controller is formed by partial derivatives in eq.(13), they are calculated as

$$\frac{\partial E_c}{\partial v} = v \qquad \frac{\partial v}{\partial F} \approx \frac{dv}{dF} \approx \frac{v(t) - v(t-1)}{F(t) - F(t-1)} \tag{23}$$

the term $\partial v/\partial F$ in eq.(13) is indirectly dependent on F and complex computation, so that an approximation can be computed by the instantaneous difference ratio. The other term $\partial F/\partial P_c$ is more tractable. So that, with F known and differentiable, a few applications of the chain rule

Algorithm 1 The modified Levenberg-Marquardt algorithm

Data: p_c or p_p, E_c or E_p, e_c or e_p
Result: p_{new}
$k := 0$; $v := 2$; $p = p_o$
$H := J^T J$; $g := J^T e\,(k)$
stop:=$(\| g \|_\infty \leq \varepsilon_1)$; $\lambda = \tau \cdot max\,(diag\,(H))$
while (not stop) and $(k < k_{max})$ and $(E\,(k) > \varepsilon_4)$ **do**
  $h = -\,(H + \lambda I)^{-1}\, g$
  if $(\| h \| \leq \varepsilon_2 \| p \|)$ **then**
    stop=true
  else
    $p_{new} = p + h$
    $\rho := \frac{\|E(k)\|^2 - \|E_{new}\|^2}{h^T(\mu h + g)}$
    if $\rho > 0$ **then**
      $p = p_{new}$
      $H := J^T J$; $g := J^T e\,(k)$
      stop=$(\| g \|_\infty \leq \varepsilon_1)$ or $\left(\| e\,(k) \|^2 \leq \varepsilon_3\right)$
      $\mu = \mu \cdot max\left(1/3, 1 - (2\rho - 1)^3\right)$; $v = 2$
    else
      $\mu = \mu \cdot v$; $v = 2v$
    end if
  end if
end while

through the five layers of the controller give the following set of learning rules.

$$\begin{aligned}
\frac{\partial F}{\partial m_{lj}^5} &= \frac{\partial F}{\partial a_j^5}\frac{\partial a_j^5}{\partial f_j^5}\frac{\partial f_j^5}{\partial m_{lj}^5} = \left(f_l^5\right)\left(1\right)\left(\frac{\sigma_{lj}^5 a_j^4}{\sum_{j=1}^r \sigma_{lj}^5 a_j^4}\right) \\
\frac{\partial F}{\partial \sigma_{lj}^5} &= \frac{\partial F}{\partial a_j^5}\frac{\partial a_j^5}{\partial f_j^5}\frac{\partial f_j^5}{\partial \sigma_{lj}^5} = \left(f_l^5\right)\left(1\right)\left(\frac{\left(\sum_{j=1}^r \sigma_{lj}^5 a_j^4\right) m_{lj}^5 a_j^4 - \left(\sum_{j=1}^r m_{lj}^5 \sigma_{lj}^5 a_j^4\right) a_j^4}{\left(\sum_{k=1}^r \sigma_{lj}^5 a_j^4\right)}\right)
\end{aligned} \tag{24}$$

The equations in (24) are the partial derivatives to update the consequent parameter values $p_{lj}^5 = \{m_{lj}^5,\ \sigma_{lj}^5\}$.

The antecedent parameter values $p_{ij}^2 = \{m_{ij}^2,\ \sigma_{ij}^2\}$ are updated when the error is propagated to the preceding layers as

$$\begin{aligned}
\frac{\partial F}{\partial p_{ij}^2} &= \frac{\partial F}{\partial a_j^5}\frac{\partial a_j^5}{\partial f_j^5}\frac{\partial f_j^5}{\partial p_{ij}^2} = \frac{\left(\sum_{j=1}^r \sigma_{lj}^5 a_j^4\right)\left(m_{lj}^5 a_j^4\right)\frac{\partial a_j^4}{\partial p_{ij}^2} - \left(\sum_{j=1}^r m_{lj}^5 \sigma_{lj}^5 a_j^4\right)\left(\sigma_{lj}^5\right)\frac{\partial a_j^4}{\partial p_{ij}^2}}{\left(\sum_{k=1}^r \sigma_{lj}^5 a_j^4\right)^2} \\
\frac{\partial a_j^4}{\partial p_{ij}^2} &= \frac{\partial a_j^4}{\partial f_j^4}\frac{\partial f_j^4}{\partial a_j^3} = \frac{\sum_{j=1}^r a_j^3 \frac{\partial a_j^3}{\partial p_{ij}^2} - a_j^3 \frac{\partial a_j^3}{\partial p_{ij}^2}}{\left(\sum_{k=1}^r a_j^3\right)^2} \\
\frac{\partial a_j^3}{\partial p_{ij}^2} &= \frac{\partial a_j^3}{\partial f_j^3}\frac{\partial f_j^3}{\partial a_{ij}^2}\frac{\partial a_{ij}^2}{\partial f_{ij}^2}\frac{\partial f_{ij}^2}{\partial p_{ij}^2} = \left(\prod_{k \neq i} a_{kj}^2\right)\left(exp(f_{ij}^2)\frac{\partial f_{ij}^2}{\partial p_{ij}^2}\right)\left(\frac{2\left(a_i^1 - m_{ij}\right)}{\left(\sigma_{ij}^2\right)^2}\right)\left(\frac{2\left(a_i^1 - m_{ij}\right)^2}{\left(\sigma_{ij}^2\right)^3}\right)
\end{aligned} \tag{25}$$

In the case of the predictor, the partial derivatives are very similar to those of the controller. Where the differential is:

$$\frac{\partial E_p}{\partial e_p} = e_p \qquad \frac{\partial e_p}{\partial v} = \alpha v \tag{26}$$

The partial derivative $\partial v/\partial p_p$ is calculated as being equal to $\partial f_l^5/\partial p_c$.

5 Walking pattern

The walking pattern generates trajectories from the design parameters (mode, sway amplitude, step time, direction) in real time. The parameters are modified at the start of each step.

5.1 ZMP and COG trajectories

The objective of defining the trajectories of the zero moment point (ZMP) and the centre of gravity (COG) is to guarantee the balance of the robot in open loop, keeping these trajectories within the base of support (BoS). The trajectory is defined in the sagittal plane, the trajectory in the coronal plane is designed in a similar way.

In the case of the ZMP, a dynamic ZMP can be represented with two equations in axes X-Y, using the simple inverted pendulum model (figure 2) and assuming a constant movement z_c of the pelvis in the transverse plane, the equation (27) is simplified. In the case of the COG, it is assumed to be identical to the centre of the pelvis.

Thus, the ZMP equation in the sagittal plane is represented as

$$x_{zmp} = \frac{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)x_i - \sum_{i=1}^{n} m_i\ddot{x}_i z_i - \sum_{i=1}^{n} I_{iy}\ddot{\omega}_{iy}}{\sum_{i=1}^{n} m_i(\ddot{z}_i + g)} \approx x_{cog} - \ddot{x}_{cog}\frac{l}{g} \tag{27}$$

where m_i is the mass, x_i, z_i is the mass position, I_i is the inertia moment and ω_i is the angular velocity. The first term (x_{cog}) is the static component and the second term ($\ddot{x}_{cog}$) is the dynamic component of the ZMP.

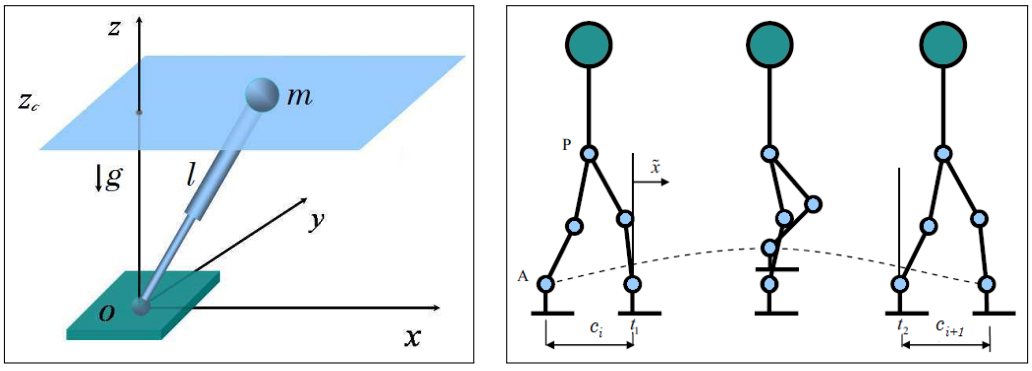

Figure 2: (Inverted pendulum model (*left*); Step parameters in the x direction (*right*)

In this paper a third-order polynomial interpolation is used to obtain the desired ZMP trajectory. A polynomial can be used to generate a trajectory in a straightforward and direct form. Moreover, the COG trajectory is designed with this trajectory.

The polynomials in eq.(28) represent the trajectories of position and velocity of the ZMP in the sagittal plane setting a normalised time to start $t_n = 0$ and end $t_n = 1$ of the step [10] without loss of generality.

$$x_{zmp}(t_n) = \sum_{t=0}^{3} b_i t_n^i \qquad \dot{x}_{zmp}(t_n) = \sum_{t=1}^{3} i b_i t_n^{i-1} \qquad b = \Omega^{-1} x_{zmp}^{boundary} \tag{28}$$

The boundary conditions $\{x_{zmp}(0),\ \dot{x}_{zmp}(0)\}$ and $\{x_{zmp}(1),\ \dot{x}_{zmp}(1)\}$ of the lifting and landing of the ZMP in each step define the polynomial coefficients in eq.(28) to get a smooth ZMP trajectory:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} x_{zmp}(0) \\ \dot{x}_{zmp}(0) \\ x_{zmp}(1) \\ \dot{x}_{zmp}(1) \end{bmatrix} \qquad \Omega b = x_{zmp}^{boundary} \tag{29}$$

By assuming that the COG x_{cog} is located to the centre of the pelvis, it is defined with the same form as (29) but with different coefficients. The boundary conditions for the COG $\{x_{cog}(0),\ \dot{x}_{cog}(0),\ x_{cog}(1),\ \dot{x}_{cog}(1)\}$ are used to design the polynomial coefficients a_i.

$$x_{cog}(t_n) = \sum_{t=0}^{3} a_i t_n^i \quad \dot{x}_{cog}(t_n) = \sum_{t=1}^{3} i a_i t_n^{i-1} \quad a = \Omega^{-1} x_{cog}^{boundary} \tag{30}$$

By substituting the x_{cog} and $\dot{x}_{cog}$ in eq.(30) and x_{zmp} in eq.(27), a direct match between ZMP and COG is obtained:

$$\sum_{t=0}^{3} b_i t_n^i = \sum_{t=0}^{3} a_i t_n^i - \left(\frac{l}{g}\right) \sum_{t=1}^{3} i a_i t_n^{i-1} \tag{31}$$

The relationship between the boundary conditions allows defining all the coefficients of the polynomials of the ZMP and the COG using (31) in a matrix form $b = \Gamma a$, where Γ is the coefficient matrix.

$$x_{zmp}^{boundary} = \Phi \Gamma \Omega^{-1} x_{cog}^{boundary} \qquad with \quad \Gamma = \begin{bmatrix} 1 & 0 & -2\frac{l}{g} & 0 \\ 0 & 1 & 0 & -6\frac{l}{g} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{32}$$

In the sagittal plane, each step can be defined by the amplitude of the feet during the double support phase before and after the single support phase (figure 2). The boundary conditions of COG are defined by with these parameters.

$$\begin{bmatrix} x_{cog}(0) & \dot{x}_{cog}(0) & x_{cog}(1) & \dot{x}_{cog}(1) \end{bmatrix}^T = \begin{bmatrix} -0.5c_i & \alpha c_i & 0.5c_{i+1} & \alpha c_{i+1} \end{bmatrix}^T \tag{33}$$

where $S_x = c_i + c_{i+1}$ is the length stride in X.

To define this trajectory, it is assumed that the position of the pelvis during a step is in the centre of the feet. The boundary conditions are parametrized by a factor α to modify the pelvis velocity. This allows acceleration at the start and end of the step and deceleration when the leg is swinging with a high value of α.

By establishing trajectories with a direct relationship, and defining boundary conditions based on the step parameters, it is simpler to define trajectories and modify them in the presence of a disturbance.

5.2 Ankle trajectory

Once the COG and ZMP trajectories are defined, it is necessary to identify the trajectories of the ankles (*support* and *swing*) according to several movement conditions, which complete the pattern.

The swinging foot trajectory ($\hat{x}_{a,swing}$) is defined by a cycloid function. It has zero velocity at the start and the end, and a maximum velocity in the swinging phase. The time (t_d) for the

double support phase (DSP) is considered. The trajectory $\hat{x}_{a,swing}$ is defined from the DSP to the other DSP, between the start time (t_1) and the final time (t_2) (figure 3 (*left*)).

$$\hat{x}_{a,swing}(t_a) = (S_x)\left(t_a - \frac{sin(2\pi t_a)}{2\pi}\right) - c_i \quad with \qquad t_a = \frac{t-(t_1+t_d/2)}{t_2-t_1-t_d} \tag{34}$$

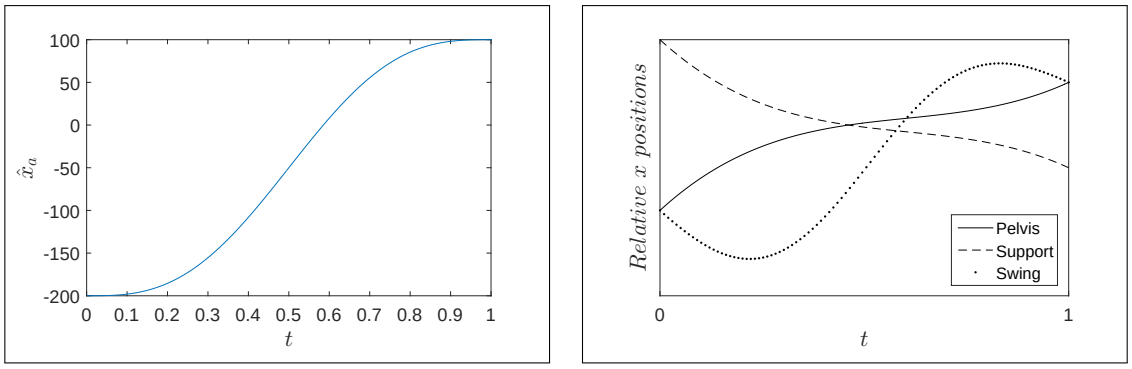

Figure 3: Position of the swinging ankle (*left*); Relative trajectories of the feet and the centre of the pelvis (*right*)

The supporting foot trajectory ($\hat{x}_{a,support}(t)$) must be the opposite action of the centre of the pelvis (x_p). These trajectories are generated in the local coordinate frame located in the front of supporting foot, and for its implementation must be transferred to the global coordinates of the robot (figure 3). In fig.(3 (*left*)) an example of the trajectories of the ankles is shown, it is assumed that the right ankle is the one supporting and the left ankle is swinging. The relative position in the global coordinate frame is:

$$x_{a,support} = -x_p(t_n) \qquad x_{a.swing} = \hat{x}_{a,swing}(t_a) - x_p(t_n) \tag{35}$$

The walking pattern is completely defined for the sagittal plane. Where the input parameters (S_x, t_1, t_2) can generate the trajectories of the ankles, knees and hip, the ZMP and the COG directly. The tuning parameters (α, t_d) are determined for the forward, turning and sideways modes.

The trajectories for the coronal plane have the same structure but with different condition boundaries.

6 Experimental results

A DARwIn-OP humanoid robot developed by the RoMeLa lab was used to implement the neuro-fuzzy system experimentally. It is 45 cm tall, weighs 2.8kg, and has 20 degrees of freedom. It has a web camera for visual feedback, and 3-axis accelerometer and 3-axis gyroscope for inertial sensing. Position-controlled Dynamixel servos are used for actuators, which are controlled by a CM730 microcontroller connected by an Intel Atom-based embedded PC at a control frequency of 100hz.

The neuro-fuzzy design begins when the fuzzy rules eq.(6) are defined incorporating information from the Darwin robot and expert knowledge. Once the structure is defined, it is necessary to reinforce learning. First, learning is applied to a realistic physical simulation to avoid damages in the real system. Finally, the neuro-fuzzy system is implemented in the Darwin robot and tested with random pushes, its performance and strategies show the advantages of the proposed scheme.

6.1 Design

The neuro-fuzzy system for the controller and the predictor are started with the same rules ($r = 9$) and with two inputs $\{\Delta\theta_x, \dot{\theta}_x\}$. But, the controller has three outputs $\{k_a, k_h, k_s\}$ and the predictor has one output $\{v\}$. So, with r defined fuzzy rules:

$$R_j : \textbf{if } \theta_x \textit{ is } \mu_{1j} \textit{ and } \dot{\theta}_x \textit{ is } \mu_{2j} \textbf{ then } k_a \textit{ is } \nu_{1j} \textit{ and } k_h \textit{ is } \nu_{2j} \textit{ and } k_s \textit{ is } \nu_{3j} \tag{36}$$

where μ_{ij} and ν_{lj} are fuzzy sets, the indexes $i = \{1, 2\}$ inputs, $l = \{1, 2, 3\}$ outputs and $j = \{1, ..., r\}$ fuzzy rules. $\{k_a, k_h, k_s\}$ are the recovery variables for the ankle eq. (4), hip eq.(5) and stepping eq.(3) strategies respectively.

With the fuzzy system defined, expert knowledge and limitations of the humanoid robot will be employed to set the initial rule base before its learning. Three membership functions are used for each input variable, and five membership functions for each output variable. The rule base is defined as follows:

Table 1: Fuzzy rule base for the feedback gains

k_a		$\Delta\theta_x$		
		NT	ZT	PT
$\dot{\theta}_x$	NV	PA	SPA	SNA
	ZV	SPA	ZA	SNA
	PV	SPA	SNA	NA

k_h		$\Delta\theta_x$		
		NT	ZT	PT
$\dot{\theta}_x$	NV	PH	SPH	SNH
	ZV	SPH	ZH	SNH
	PV	SPH	SNH	NH

k_s		$\Delta\theta_x$		
		NT	ZT	PT
$\dot{\theta}_x$	NV	NS	ZS	SPS
	ZV	SNS	ZS	SPS
	PV	SNS	ZS	PS

The membership functions in table 1 are defined using expert knowledge and the minmax ranges for each push recovery strategy. The term sets of the inputs and output variables and their limits are selected as follows:

$$\begin{aligned}
T(\theta_x) &= \{\mu_{11}, \mu_{12}, \mu_{13}\} = \{NT, ZT, PT\} && \{-45, 45\} \\
T(\dot{\theta}_x) &= \{\mu_{21}, \mu_{22}, \mu_{23}\} = \{NV, Z, PV\} && \{-120, 120\} \\
T(k_{xa}) &= \{\nu_{11}, \nu_{12}, \nu_{13}, \nu_{14}, \nu_{15}\} = \{NA, SNA, ZA, SPA, PA\} && \{-0.35, 0.35\} \\
T(k_{xh}) &= \{\nu_{21}, \nu_{22}, \nu_{23}, \nu_{24}, \nu_{25}\} = \{NH, SNH, ZH, SPH, PH\} && \{-1, 1\} \\
T(k_{xs}) &= \{\nu_{31}, \nu_{32}, \nu_{33}, \nu_{34}, \nu_{35}\} = \{NS, SNS, ZS, SPS, PS\} && \{-0.85, 0.85\}
\end{aligned} \tag{37}$$

where the letters for each fuzzy set are: Negative (N), Small Negative (SN), Zero (Z), Small Positive (SP), Positive (P), Trunk (T), Velocity (V), Ankle (A), Hip (H) and Step (S). For example, SPA meaning Small Positive Ankle.

6.2 Learning

The neuro-fuzzy system learns from random pushes and during walking and is able to successfully absorb multiple pushes. Simulation trials were performed for 3 push types (frontal, behind and lateral) with 6 repetitions in different strength ranges (small, medium, large). Reinforcement learning in the simulation was applied to 54 trials resulting in the following fuzzy surface fig.(4), for each output gain K.

The ankle gain surface shows that the strategy is applied in all ranges, with a large or medium push. Its gain is high, but with a small push it is regulated. The hip strategy is applied when

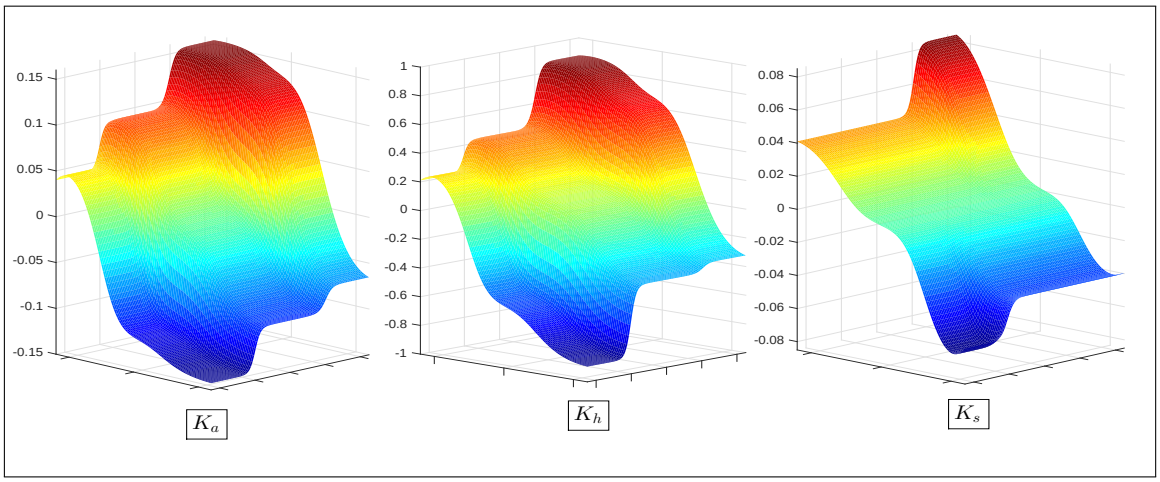

Figure 4: Ankle feedback surface after simulation learning (*left*); Hip feedback surface after simulation learning (*middle*); Step feedback surface after simulation learning (*left*)

the disturbance is greater, allowing a recovery without having to execute a step. Finally, the step strategy surface shows boundaries where it is inevitable to apply the step, but inside the robot kinematic.

6.3 Results

After designing and training the neuro-fuzzy system in a physically realistic simulation, the system was implemented in the Darwin robot. A trial was defined with multiple pushes to evaluate the ability of the neuro-fuzzy system to represent expert knowledge and respecting the limits, as well as assess the reinforcement learning algorithm to update the system.

Figure (5) shows an experimental test with 6 random pushes in walking forward mode, The first and second plots show θ_x and $\dot{\theta}_x$, respectively. The unknown strength push is frontal ($3s$,$29.9s$ and $40s$) and behind ($9s$, $15.5s$ and $21.6s$) are marked with vertical dash lines in the plots. The first push is the strongest requiring application of the three strategies as shown in the fig.(6), where the robot steps forward and corrects the trunk position angle to an upright position. The feedback gains are weighted using the effect of the inferior strategy learnt in the simulation trails respecting the limits defined in the design.

Note the sudden velocity change at the moment of pushes. Then, the robot is able to recover its velocity to the normal level and maintain walking. This is because the external reinforcement signal is applied with minimum delay to the neuro-fuzzy system, and this has a low computational cost with 18 fuzzy rules and 64 parameters to update. In a comparison between the Levenberg Marquardt algorithm and the backpropagation algorithm. The proposed algorithm has an average time of 0.5 ms with 8 average iterations, while, the backpropagation has an average time of 4 ms with 80 average iterations. This allows efficient implementation in embedded systems.

Finally, a phase plot between θ_x and $\dot{\theta}_x$ is shown in fig. (7), it shows the theoretical boundaries for each push recovery strategy using a lineal inverted pendulum model. The boundaries are calculated using the following parameters: $z_c = 0.295$, $d = 0.05$, $T_H = 0.3$, $m = 2$, $g = 9.81$,

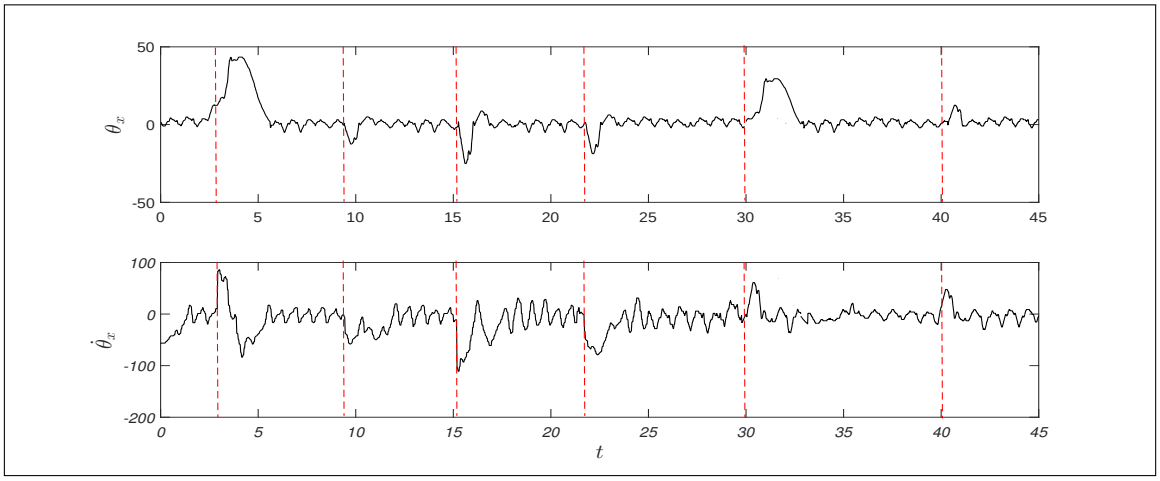

Figure 5: Experimental test with 6 random pushes without failure

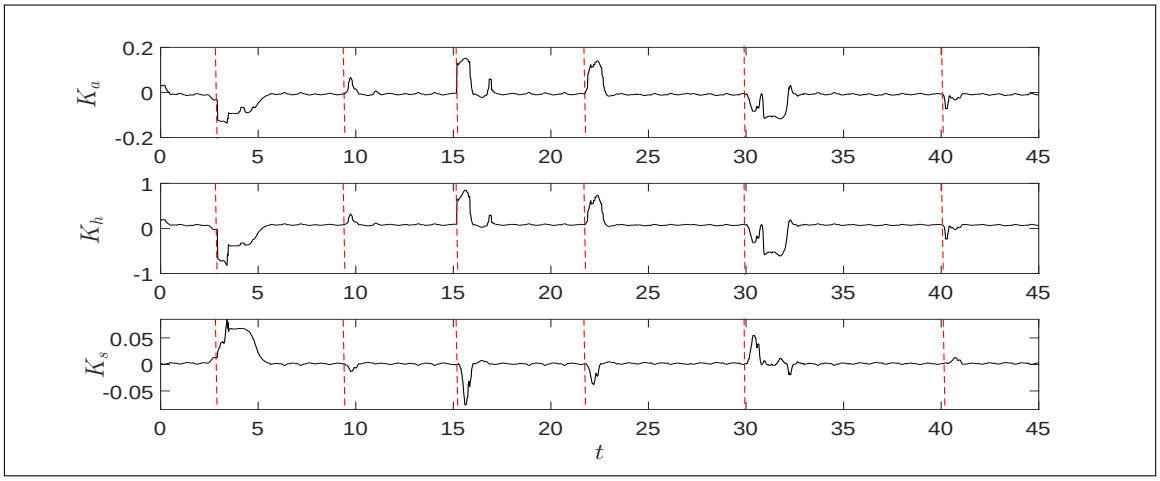

Figure 6: Gains of the 3 push recovery strategies with 6 random push after learning

$w = \sqrt{g/z_c}$, $\tau_{hip}^{max} = 1$ and $x_{capture}^{max} = 0.08$ [17].

$$\begin{aligned} Ackle: &\quad \left|\frac{\dot{\theta}_x}{w} + \theta_x\right| < \frac{d}{z_c} \\ Ankle + Hip: &\quad \left|\frac{\dot{\theta}_x}{w} + \theta_x\right| < \frac{d}{z_c} + \frac{\tau_{hip}^{max}\left(e^{-wT_H}-1\right)^2}{mgz_c} \\ Ankle + Hip + stepping: &\quad \left|\frac{\dot{\theta}_x}{w} + \theta_x\right| < \frac{d}{z_c} + \frac{\tau_{hip}^{max}\left(e^{-wT_H}-1\right)^2}{mgz_c} + \frac{x_{capture}^{max}}{z_c} \end{aligned} \tag{38}$$

The phase plot shows that the neuro-fuzzy system improved the robustness to strong pushes with

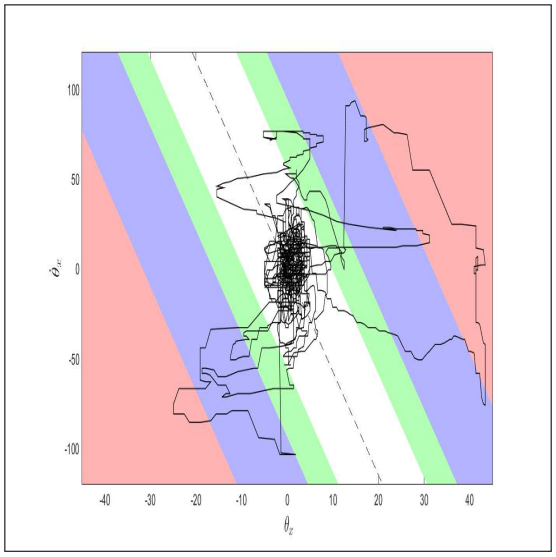

Figure 7: The phase plot shows how the neuro-fuzzy system responds to 6 unknown pushes. The coloured areas are the theoretical boundaries for the ankle(white), hip (light green), step (light blue) and unstable (light red).

a larger stable region. So, The robot already learnt the right ankle, hip and stepping strategies to cope with pushes from any direction.

7 Conclusions

In this paper, a practical method to implement a full body push recovery system on a general humanoid robot without specialised sensor and actuators is proposed. The method consists of a neuro-fuzzy system designed by combining expert knowledge, system constraints and online reinforcement learning.

This approach has a number of advantages over previous approaches. It is able to recover from unknown disturbances from any direction performing a combination of push recovery strategies based on the current system state without an accurate dynamical model of the robot or environment.

A straightforward and robust online reinforcement learning algorithm updates the parameters for the neuro-fuzzy controller and predictor to improve the push recovery performance using both simulation environment as well as on a small-size humanoid. Implementation using the modified LM algorithm guarantees low-cost computation.

It is integrated with a non-periodic, step-omnidirectional walking generator where step parameters produces soft curves and can be flexibly changed as a response to a disturbance. Moreover, it does not attempt to follow a future ZMP reference, only the joint trajectories are prescribed using the step parameters and a simple inverted pendulum model.

Experimental results show that the trained system can successfully develop a full body push recovery under external perturbations while walking in an arbitrary direction. There is a significant improvement of the stable region in the sagittal plane from a low number of experiences, but enough for the neuro-fuzzy controller to produce convincing push recovery capabilities.

Possible future work includes incorporating strategies to unreliable terrains and robustness.

Acknowledgment

Thanks to the support of the Laboratory of Humanoid Robots at PCC-UNAM for the facilities provided. To Dr. Fernando Arambula Cosio and Dr. Mario Pena Cabrera for the facilities and knowledge provided during the development of the project.

Bibliography

[1] Adiwahono A.H., Chew C.M., Liu B. (2013); Push recovery through walking phase modification for bipedal locomotion, *International Journal of Humanoid Robotics*, DOI: http://dx.doi.org/10.1142/S0219843613500229, 10(3), 1350022, 2013.

[2] Allgeuer P., Behnke S.(2014); Fused angles for body orientation representation, *Proceedings of 9th Workshop on Humanoid Soccer Robots of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, DOI: 10.1109/IROS.2015.7353399, 366-373, 2014.

[3] Faber F., Behnke S. (2007); Stochastic optimization of bipedal walking using gyro feedback and phase resetting, *7th IEEE-RAS International Conference on Humanoid Robots*, 203-209, doi: 10.1109/ICHR.2007.4813869, 2007.

[4] Fuller R. (2013); *Introduction to neuro-fuzzy systems*, Vol. 2, Springer Science and Business Media, 2013.

[5] Hyon S.H., Osu R., Otaka Y. (2009); Integration of multi-level postural balancing on humanoid robots, *IEEE Int. Conf. Robotics and Automation (ICRA)*, doi: 10.1109/ROBOT.2009.5152434, 1549-1556, 2009.

[6] Komura T., Leung H., Kudoh S., Kuffner J. (2005); A feedback controller for biped humanoids that can counteract large perturbations during gait, *IEEE Int. Conf. Robotics and Automation (ICRA), Barcelona, Spain*, 1989-1995, 2005.

[7] Koolen T., de Boer T., Rebula J.R., Goswami A., Pratt J.E. (2012): Capturability-based analysis and control of legged locomotion: Theory and application to three simple gait models - Part 1, *Int. J. of Robotics Research*, 31(9), 1094-1113, 2012.

[8] Missura M., Behnke S. (2015); Gradient-Driven Online Learning of Bipedal Push Recovery, IEEE/RSJ, *Int. Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, doi: 10.1109/IROS.2015.7353402, 387-392, 2015.

[9] Missura M., Behnke S. (2013); Omnidirectional Capture Steps for Bipedal Walking, *The IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 14-20, doi: 10.1109/HUMANOIDS.2013.7029949, 2013.

[10] Park J.-W., Kim J.-Y., Oh J.-H. (2008); Online Walking Pattern Generation and Its Application to a Biped Humanoid Robot - KHR-3 (HUBO), *Advanced Robotics*, 22(2), 159-190, 2008.

[11] Pratt J., Carff J., Drakunov S., Goswami A. (2006); Capture point: A step toward humanoid push recovery, *IEEE-RAS Int. Conf. Humanoid Robots*, Genova, Italy, 200-207, 2006.

[12] Pratt J.E., Koolen T., de Boer T., Rebula J.R., Cotton S., Carffer J., Johnson M., Neuhaus P.D. (2012); Capturability-based analysis and control of legged locomotion: Application to M2V2, a lower-body humanoid - Part 2, *Int. J. of Robotics Research*, 31(10), 1117-1133, 2012.

[13] Ranganathan A. (2004), The Levenberg-Marquardt Algorithm, *Tutorial on LM algorithm*, 2004.

[14] Rebula J., Pratt J., Canas F., Goswami A. (2007); Learning capture point for improved humanoid push recovery, *IEEE-RAS Int. Conf. Humanoid Robots, Pittsburgh, PA*, doi: 10.1109/ICHR.2007.4813850, 65-72, 2007.

[15] Semwal V.B., Chakraborty P., Nandi G.C. (2015); Less computationally intensive fuzzy logic (type-1)-based controller for humanoid push recovery. *Robotics and Autonomous Systems*, 63, 122-135, 2015.

[16] Si J., Wang Y.T. (2001); On-line Learning Control by Association and Reinforcement, *IEEE Trans. on Neural Networks*, doi: 10.1109/72.914523, 12(2), 264-276, 2001.

[17] Stephens B. (2007); Humanoid push recovery, *IEEE-RAS Int. Conf. Humanoid Robots*, IEEE Press, Pittsburgh, PA, 589-595, 2007.

[18] Tedrake R. (2004); Stochastic policy gradient reinforcement learning on a simple 3d biped, *Proc. of the 10th Int. Conf. on Intelligent Robots and Systems*, 2849-2854, 2004.

[19] Wieber P.B. (2006); Trajectory free linear model predictive control for stable walking in the presence of strong perturbations, *IEEE-RAS Int. Conf. Humanoid Robots*, IEEE Press, Nashville, doi: 10.1109/ICHR.2006.321375, 137-142, 2006.