Approximate optimal control for reaching and trajectory planning in a humanoid robot

S. Ivaldi, M. Fumagalli, F. Nori, M. Baglietto, G. Metta, G. Sandini

***Abstract*— Online optimal planning of robotic arm movement is addressed. Optimality is inspired by computational models, where a "cost function" is used to describe limb motions according to different criteria. A method is proposed to implement optimal planning in Cartesian space, minimizing some cost function, by means of numerical approximation to a generalized nonlinear model predictive control problem. The Extended RItz Method is applied as a functional approximation technique. Differently from other approaches, the proposed technique can be applied on platforms with strict control temporal constraints and limited processing capability, since the computational burden is completely concentrated in an off-line phase. The trajectory generation on-line is therefore computationally efficient. Task to joint space conversion is implemented on-line by a closed loop inverse kinematics algorithm, taking into account the robot's physical limits. Experimental results, where a 4DOF arm moves according to a particular nonlinear cost, show the effectiveness of the proposed approach, and suggest interesting future developments.**

I. INTRODUCTION

It is a common belief that the human body moves "optimally" and that feedback control is performed during human reaching movements [1]. According to computational motor control, human limbs trajectories during goal-directed movements can be modeled by an optimization process guided by suitably defined performance indexes [2]: more precisely, as the minimization of nonlinear cost functions, sometimes even non-differentiable [3]. In literature different computational models can be found, describing trajectories as the minimization of variance [6], torque change [7], jerk [5], energy of moto-neurons signals [4]. In humanoid robotics, where reaching is the fundamental action primitive, such models are particularly interesting [8]. During such movements (e.g. reaching, walking) the focus is not only on the successful reach, but also on the trajectory performed by limbs: by implementing computational models on the robotic platform it is possible to mimic human movements, and achieve, within certain approximations, human-like behaviors [11], [12]. In this perspective, the robot must be provided with a tool that is able to plan "optimally": once given the biologically inspired principles, generate the trajectories and execute the desired motions in real-time (possibly without being resource-demanding). Unfortunately, implementations on humanoid platforms (e.g., see [13], [14], [9], [10]) face notable computational limits. Rather than finding a generalized optimal solution to the planning problem, which would surely incur into the curse of dimensionality and prevent the application in real-time, the proposed approaches in literature focus on the optimization of single point-to-point movements. The corresponding optimal control problems are usually tackled via numerical methods and nonlinear programming algorithms, but the optimization process requires heavy computations and prevents the application in real-time :as an example, in [10] a single movement generation is reported to take from 1 to 4 minutes, even with a fast optimizer as IPOPT [28].

Since closed-form solutions are utterly hard to find (impossible in many cases), approximate solutions can be addressed. In particular, Nonlinear Model Predictive Control (NMPC) methods can be used. The explicit precomputation of NMPC is prohibitive[1] for state/parameters above $\mathbb{R}^{10}$: e.g. in [26] a fast direct multiple shooting algorithm and several approximations were made to reduce a 20 CPU seconds computations on a 3GHz Pentium IV to $200ms$, for a 5 state 0.15 seconds trajectory.

Among the possible options, here an off-line approximation of the global control law is preferred: the complete precomputation of a neural approximation of an explicit Finite/Receding Horizon (FH/RH) optimal control law (supported by an intermediate control to compensate modeling errors) allows finding almost instantly the controls, in the order of μs, leaving the machine free for other tasks during on-line execution (e.g. contact detection, learning, etc.). Note that the closed-loop control on the robot considered in this work runs at $5 - 20ms$ (at least $5ms$ are necessary for force control, which will be used with this method in the future).

Note also that the aforementioned techniques usually solve single optimization problems, i.e. each trajectory is the result of an optimization problem (typically varying its boundary conditions); conversely, in the proposed approach a generalized solution is found, for all the possible initial/desired conditions (positions, velocities and so on). Thus, in the on-line phase no further processing is required; the computation of the on-line controls is very fast, consisting only in a single forward pass of a neural network, as will be discussed later on; real-time performances can be guaranteed; furthermore, the machine controlling the robot does not require an external optimization routine (usually resource consuming), or

G. Metta, M. Baglietto and G. Sandini are with the Department of Communication, Computer and System Sciences, Faculty of Engineering, University of Genoa, Italy.

S. Ivaldi, M. Fumagalli, F. Nori, G. Metta and G. Sandini are with the Robotics, Brain and Cognitive Sciences Department, Italian Institute of Technology, Genoa, Italy. (corresponding author phone: +39-010-71781453; e-mail:serena.ivaldi@iit.it).

[1]The reader should see [27], where off-line precomputation, delay compensation and other techniques were surveyed, discussing reasonable compromises between computational time, convergence of the method, approximation performances and real-time guarantee.

licensed software, nor specific hardware.

The proposed technique consists of two steps. In the first, off-line, a suitable sequence of approximating functions is trained, so that they can approximate the sequence of optimal control functions of a stochastic Finite Horizon problem. The ERIM is chosen as a functional approximation technique, while the use of feed-forward neural networks guarantees that the optimal solutions can be approximated at any desired degree of accuracy [21], [17]. Planning is carried out in Cartesian space and once the optimal trajectories are determined, they are converted into motor commands – joint level – using a closed-loop inverse kinematics algorithm (which takes into account the manipulator physical limitations). In the on-line phase, a single forward computation of a neural network (consisting of few elementary operations such as additions, multiplications etc. - see [23]) yields the proper control at each time instant. The feasibility of this approach has already been tested on the control of a thrusts-actuated non-holonomic robot [18], while numerical results showing its effectiveness for different cost functions were presented in [22], for the motion of a simpler planar manipulator.

The remainder of the paper is organized as follows. Section II describes the robotic setup: the humanoid robot James. Sections III and IV describe the FH and RH neural controllers, along with the training procedure and the closed loop inverse kinematics, respectively. Sections V shows experimental results. Section VI discusses future works, while Section VII draws the conclusions.

II. ROBOTIC SETUP

James [30] is a humanoid torso, consisting of a head, a left arm and hand, with the overall size of a 10 years old boy (see Fig. 1). Among the 7 DOF of the arm, only 4 have been considered, i.e. 3 for the shoulder and 1 for the elbow. In Tab. I the range of the four joints positions is reported. Torque is transmitted to the joints by rubber toothed belts, pulleys and stainless-steel tendons, actuated by rotary DC motors. Motors are controlled with 12 Digital Signal Processing (DSP) boards (Freescale DSP56F807, 80MHz, fixed point 16 bits). Each board provides for the regulation of two motors, with a 1KHz control loop. Optical encoders are used for the feedback position control loop implemented on the boards. A CAN-bus line allows the communication between the boards and the remote PC, where an ESD CAN-USB is provided. Reference position and velocity commands can be set by the user through the latter.

DSP boards have limited memory and computation capability and cannot support more than simple operations, namely low level motor control (basically PID position control), signal acquisition and pre-filtering from the optical encoders. For this reason, implementing an on-line controller directly on the DSP boards is impossible in the current setup: the control is indeed computed on the remote PC, while the DSP boards act like mere low level controllers.

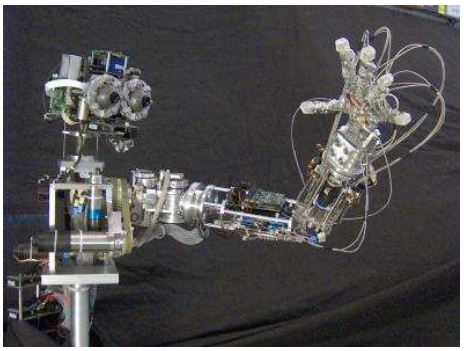

Fig. 1. The humanoid robot James.

$q_0[^\circ]$	$q_1[^\circ]$	$q_2[^\circ]$	$q_3[^\circ]$	
-10	-140	-115	0	min
150	100	30	100	max

TABLE I

VALUE RANGES OF THE ARM JOINT POSITIONS.

III. OPTIMAL TRAJECTORY PLANNING THROUGH FINITE/RECEDING HORIZON REGULATORS

A. Problem formulation

The elements involved in the planning phase are the end-effector of the manipulator and the target. Their Cartesian coordinates, with respect to a fixed reference frame, at time instant t are respectively denoted[2] by $x_t^r, x_t^g \in \mathbb{R}^3$ (each of them consisting of three elements, in the x, y, z direction: the orientation is neglected for the moment, but the approach can be easily generalized for the orientation problem). Consider ξ_t a generic vector describing the difference between the current end-effector coordinates and the target ones, $\xi_t \triangleq [x_t^g - x_t^r]$, or a more detailed vector containing position, velocities, acceleration errors, etc. (e.g. $\xi_t \triangleq [x_t^g - x_t^r, \dot{x}_t^g - \dot{x}_t^r]$). It is assumed that the following compact model can be used to describe the evolution of the end-effector with respect to the target ξ_t, if the control u_t acts on the robot:

$$\xi_{t+1} = f(\xi_t, u_t) \ , \ t = 0, 1, \ldots \tag{1}$$

where at the time instant t, ξ_t is the state vector, taking values from a finite set $\Xi \subseteq \mathbb{R}^n$, and u_t is the control vector, constrained to take values from a finite set $U \subseteq \mathbb{R}^m$. At any time instant t, the desired state is $\xi_t^* = 0$, meaning that the goal is to bring the difference vector between the end-effector and the target to zero. Thanks to the time invariance of the system dynamics and of the cost function, $t = 0$ can be considered as a generic time instant. Then, a single (functional) FH optimization problem is addressed.

Problem 1 (Reaching): Find a sequence of optimal control functions $\mu_0^\circ, \ldots, \mu_{N-1}^\circ$, that minimize the cost func-

[2]Hereinafter, $x(t) = x_t, u(t) = u_t$, where t is a generic time instant. The subscript i and t will be used referring to a RH or FH sequence of states/controls.

tional

$$\bar{\mathcal{J}} = \underset{\xi_0 \in \Xi}{E} \left\{ \sum_{i=0}^{N-1} h_i(\xi_i, \mu_i(\xi_i)) + h_N(\xi_N) \right\} \qquad (2)$$

subject to the constraints $\mu_i^\circ \in U \subseteq \mathbb{R}^m$ *and* $\xi_{i+1} = f(\xi_i, \mu_i(\xi_i))$.

The i-th couple "controller-function" of the FH problem is shown in Fig. 2. It is remarkable that the problem is generalized for every possible initial state ξ_0 (i.e. for every possible robot and target's coordinates) in virtue of the Expectation operator E.

The solution of Prob. 1, which is a FH problem, allows to solve without additional effort the corresponding RH problem. In fact, by making the assumption that the desired state $\xi_t^* = 0$ holds for N stages, a *certainty equivalence principle* is implicitly applied: at time instant t, the target vector ξ_t^* is supposed to remain constant for N time instants, that is: $\xi_{t+i+1}^* = \xi_{t+i}^*, i = 0, \ldots, N-1$. If at each time instant a FH problem is solved and only the first control function is retained, then it is possible to exploit the generalized FH solution also in the RH case.

Remark 1: This is the situation corresponding to the tracking of unknown/unpredictable targets: the robot assumes the target to retain the current position, plans a trajectory and moves toward it. A change in the target's position does not affect the tracking behavior, since at each time instant the target's coordinates are measured and the control corresponding to a newer trajectory is performed. Thus, the method can be applied to both static or moving targets.

The corresponding RH problem is:

Problem 2 (Tracking): For every time instant $t \geq 0$*, find the RH optimal control law* $u_t^\circ = \mu_t^\circ(\xi_t) \in U$*, where* μ_t° *is the first control function of the sequence* $\mu_{0|t}^\circ, \ldots, \mu_{N-1|t}^\circ$ *that minimize the FH cost functional*

$$\bar{\mathcal{J}}_t = \underset{\xi_t \in \Xi}{E} \left\{ \sum_{i=0}^{N-1} h_i(\xi_{t+i}, \mu_{i|t}(\xi_{t+i})) + h_N(\xi_{t+N}) \right\}. \qquad (3)$$

μ_0° is a time invariant control function, i.e. $u_t^{RH} = \mu^{RH}(\xi_t) = \mu_0^\circ(\xi_t)$.

Remark 2: The goal "to reach a target" could also be expressed by a hard constraint like $\xi_N = 0$, rather than a soft one defined by function h_N. The latter has been preferred, since the robot could not be able to reach the target perfectly as a consequence of the unpredictable behavior of the target or the robot's intrinsic physical limits; for example, the target could be outside the arm's reachable space. In such situations, an optimization problem with a hard constraint would fail (i.e. no solution). A convex cost term like h_N instead, always ensures a solution.

Remark 3: Note that the solution of a FH/RH problem for a certain $\xi_0 = \hat{\xi}$ yields a sequence of controls and points discretizing the optimal trajectory. Solving for all the possible and admissible $\xi_0 \in \Xi$ means that a set of L sequences of controls are computed, for all the possible initial states. This set is used as a training set for the neural networks, so that through the (off-line) learning procedure they can approximate the optimal control functions. Then, each single control in real-time does not require any optimization process, but is immediately retrieved from the approximated "global" solution.

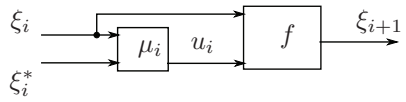

Fig. 2. The i-th element couple, when the control task is "unfolded" in time and a desired value ξ_i^* is considered. The input to the neural controller can be the difference between the desired and the current state, only if f is linear: in that case this solution is not only feasible, but preferred since it halves the inputs to the neural controller, thus reducing the computational complexity of the method. In a general situation (e.g. f nonlinear), the inputs are both ξ_i and ξ_i^*.

Remark 4: With reference to Fig. 2, it is worthy to note that the proposed method is particularly appealing if function f is a nonlinear dynamic model of the robot, i.e. the function $q_{i+1} = f(q_i, \dot{q}_i)$ is known. In this case, the intermediate Closed-Loop Inverse Kinematic control (CLIK), described later on, would not be necessary as the control law could take into account the Inverse Kinematics and compute the controls in the joint space directly. Such function has not been estimated yet, thus the Inverse Kinematics problem must be solved taking into account the imperfect discretization of all the joint variables. More details are in Section IV. Note that if $q_{i+1} = f(q_i, \dot{q}_i)$ is known, since the kinematics of the robot is known, it is straightforward to find $[q_{i+1}, x_{i+1}] = F(q_i, x_i, \dot{q}_i, \dot{x}_i)$.

B. From a functional optimization problem to a nonlinear programming one

Since the RH problem exploits the solution of the FH problem, only the latter is discussed hereinafter. In order to solve Problem FH the ERIM [16] is applied: the admissible control functions $\mu_0, \mu_1, \ldots, \mu_{N-1}$ are constrained to take the form of one-hidden-layer (OHL) neural networks:

$$\hat{\mu}_i(\xi_i, w_i) = col\left(\sigma \left[\sum_{h=1}^{\nu} c_{ihj} \varphi_h(\xi_i, \kappa_{ih}) + b_{ij} \right] \right) \qquad (4)$$

with $j = 1, \ldots, m$, where the vector , $w_i \in \mathbb{R}^W$, being $w_i = \{c_{ihj}, \kappa_{ih}, b_{ij}, \forall h, j\}$ and $W = (n+1)\nu + m(\nu+1)$, collects all the parameters to be optimized in the i-th network. $\hat{\mu}_i(\cdot, w_i) : \mathbb{R}^n \times \mathbb{R}^W \mapsto \mathbb{R}^m$, $c_{hj}, b_j \in \mathbb{R}, \kappa_h \in \mathbb{R}^{n+1}$, being ν the number of *neurons* constituting the network; $\sigma(\cdot) = U \tanh(\cdot)$ bounds the controls within the admissible range $[-U, U]$; φ is a basis function (i.e. a sigmoid). Input and output normalization are assumed, but not indicated for the clarity of notation. By substituting (4) into (2), the

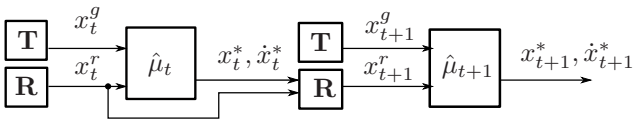

Fig. 3. Unfolding in time of the approximating neural networks. **T** accounts for the "target", while **R** for the "robot".

general functional cost $\bar{\mathcal{J}}(\mu_0, \mu_1, \ldots, \mu_{N-1})$ is turned into a function $\hat{\mathcal{J}}_\nu(w)$ which is only dependent on a finite number of real parameters, $w = col(w_i, i = 0, 1, \ldots, N-1)$. Prob. 1 is thus restated as:

Problem 3 (Neural Reaching): Find the optimal vectors of parameters $w_0^\circ, \ldots, w_{N-1}^\circ$ that minimize the cost function

$$\hat{\mathcal{J}}_\nu = \underset{\xi_0 \in \Xi}{E} \left\{ \sum_{i=0}^{N-1} h_i(\xi_i, \hat{\mu}_i(\xi_i, w_i)) + h_N(\xi_N) \right\}$$

subject to the constraints $\hat{\mu}_i(\xi_i, w_i) \in U \subseteq \mathbb{R}^m$ and $\xi_{i+1} = f(\xi_i, \hat{\mu}_i(\xi_i, w_i))$.

Then, for every time instant t, the time-invariant RH control law corresponds to $u_t^{RH} = \hat{\mu}^{RH}(\xi_t, w_0^\circ) = \hat{\mu}_0^\circ(\xi_t, w_0^\circ)$.

C. Solution of the nonlinear programming problem by stochastic gradient

The optimal parameters in the OHL control functions can be found by a stochastic gradient steepest descent:

$$w_i(k+1) = w_i(k) - \alpha(k) \nabla_{w_i} \hat{\mathcal{J}}_\nu [w(k)] + \\ + \eta(w_i(k) - w_i(k-1)) \qquad (5)$$

for $k = 0, 1, \ldots$, with the usual regularization term $\eta \in [0, 1]$. The convergence of the method is assured by a particular choice of the step size $\alpha(k)$, that must fulfill a set of conditions [20]. The partial derivatives necessary for the application of (5): $\frac{\partial \hat{\mathcal{J}}_\nu}{\partial w_i} = \frac{\partial \hat{\mathcal{J}}_\nu}{\partial u_i} \frac{\partial \hat{\mu}_i(\xi_i, w_i)}{\partial w_i}$. are computed in two steps. In a *forward phase* the system and the neural controllers are unfolded in time, making a chain where the feedback is explicit (see Fig. 3) (at iteration step k, given the initial state ξ_0, all the intermediate states ξ_i and controls u_i generated by the sequence of OHL networks are computed). In a *backward phase*, all the gradient components are computed and "back-propagated" through the networks' chain. The recursive propagation is described by the following equations, for $i = N-1, N-2, \ldots, 0$:

$$\frac{\partial \hat{\mathcal{J}}_\nu}{\partial u_i} = \frac{\partial h_i(\xi_i, u_i)}{\partial u_i} + \frac{\partial \hat{\mathcal{J}}_\nu}{\partial \xi_{i+1}} \frac{\partial f(\xi_i, u_i)}{\partial u_i}$$

$$\frac{\partial \hat{\mathcal{J}}_\nu}{\partial \xi_i} = \frac{\partial h_i(\xi_i, u_i)}{\partial \xi_i} + \frac{\partial \hat{\mathcal{J}}_\nu}{\partial \xi_{i+1}} \frac{\partial f(\xi_i, u_i)}{\partial \xi_i} + \frac{\partial \hat{\mathcal{J}}_\nu}{\partial u_i} \frac{\partial \hat{\mu}_i(\xi_i, w_i)}{\partial \xi_i}$$

initialized by $\partial \hat{\mathcal{J}}_\nu / \partial \xi_N = \partial h_N(\xi_N) / \partial \xi_N$.

IV. CLOSED LOOP INVERSE KINEMATICS

The arm motion control focuses on the first four joints, i.e. three joints of the shoulder and one of the elbow (for further details see Section II). James hand is considered as the end effector of the manipulator. As already pointed out, the rotation of the hand is neglected and will be object of future works. Denoting with $x^r = [x, y, z]$ the Cartesian coordinates of the end effector with respect to a fixed reference frame, and with $q = [q_0, q_1, q_2, q_3]$ the vector of the joint position variables of the arm (see Tab. I), then the forward kinematics $x^r(t) = f_{\text{arm}}(q(t))$, $f_{\text{arm}} : \mathbb{R}^m \to \mathbb{R}^n$ is found using the Denavit-Hartenberg convention [19]. The target's Cartesian coordinates are denoted by $x^g(t)$. The orientation of the end effector is neglected. Within this context, the 4 DOF manipulator is redundant.

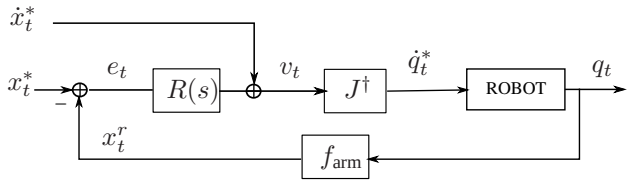

Fig. 4. A simple CLIK scheme. The block $J^\dagger$ refers to (6). The retrieving of the target's cartesian coordinates is not modeled, as it would require to discuss the robotic visual system, the target identification module etc. For the sake of simplicity, many details about the closed loop control are voluntarily neglected, to keep the scheme clear.

Given a desired trajectory for the end-effector, denoted by $x^*(t)$, and its velocity profile, $\dot{x}^*(t)$, in the Cartesian space, Cartesian and joint space velocity commands, $v(t)$ and $q^*(t)$ respectively, are computed with a Closed Loop Inverse Kinematic (CLIK) algorithm, as shown in Fig. 4.

Among the possible ways to invert the kinematics the following is used (see Fig. 4):

$$\dot{q}^*(t) = J^\dagger v(t) + (I - J^\dagger J)\dot{q}_a(t) \\ = J^\top (JJ^\top + k^2 I)^{-1} v(t) + (I - J^\dagger J)\dot{q}_a(t) \qquad (6)$$

where $J^\dagger$ is a damped least-squares pseudo-inverse of the Jacobian J; $\dot{q}_a(t)$ represents an arbitrary joint velocity vector which is projected in the null-space of the Jacobian matrix by the projector $(I - J^\dagger J)$ (I is the identity matrix). $\dot{q}_a(t)$ was chosen such that joints positions were maintained far from their physical limits [29]. In such manner it is possible to cope with singularities and to exploit the intrinsic redundancy of the manipulator. The parameter k^2 in (6) is determined adaptively as in [25], i.e. depending on the smallest singular value $\sigma_{\min}$ of the Jacobian matrix:

$$k^2 = \begin{cases} 0 & \sigma_{\min} > \bar{\sigma} \\ [1 - (\frac{\sigma_{\min}}{\bar{\sigma}})^2]\bar{k} & \sigma_{\min} < \bar{\sigma} \end{cases}. \qquad (7)$$

In this specific case, $\bar{k} = 0.10$ and $\bar{\sigma} = 0.20$, the latter determined sperimentally. The parameters were chosen by driving the arm to singular positions.

Note that the singularity is solved by acting on the singular values, which are configuration-dependent, while redundancy is resolved by selecting the solution which stays furthest away from the joints bounds. A point-wise approach like this may not lead to the best solution for the overall trajectory. It is reasonable to solve the Inverse Kinematics in this way, since "globally" the control functions are already an approximation of the global ones, and the smoothing properties of the neural networks should prevent rough behaviors.

In order to avoid the "drift" effect due to the discretization of the joints positions (see [19]) a closed loop inverse kinematic algorithm is used. The classical scheme relies on a purely proportional regulator, i.e. $R(s) = K_e$, where K_e is a diagonal positive defined matrix. In an ideal situation, the correction term $K_e e(t)$, where

$$e(t) = x^*(t) - x^r(t) \ , \qquad (8)$$

guarantees convergence to zero of the cartesian error and the error dynamic $\dot{e} + K_e e = 0$ is asymptotically stable. The convergence velocity of such system depends on the eigenvalues of the gain matrix $K_e > 0$ [29][19].

In order to perform tracking tasks, the discrete-time system should respond to rapid variation of the tracking trajectory: hence, the gain K_e should be raised until some reasonable performances are met. Unfortunately, K_e cannot be raised *ad libitum*: its upper-limit is determined by the physics of the problem. Moreover, high-frequency terms (e.g. unmodeled dynamics) also prevent to raise it to a sufficient value: in the experiments, a too high valued K_e (e.g. $K_e = 20I$) was already revealing elastic effects (a step-response was used to investigate the system); higher (e.g. $K_e = 40I$) was compromising the safety of the robot.

For these reasons the simple proportional gain K_e was substituted with the following:

$$R(s) = K_e \frac{1 + s\tau_e}{s} \tag{9}$$

where K_e is still a positive diagonal matrix, τ_e a time constant. Incidentally, (9) corresponds to a PI controller, where the proportional gain is $K_e\tau_e$ and the integral one is K_e. The sampling time $\Delta t = 5 - 50ms$ is then fundamental for the proper tuning of the digital integral gain; the time constant τ_e (corresponding to the zero $-1/\tau_e$) can be thus properly raised. For example, $\tau_e = 30, K_e = 0.5$ when $\Delta t = 20ms$. It is worth noting that the use of this regulator was necessary to have high proportional gain and without incurring in an instability of the system. With the pure proportional K_e, a high gain was compromising the transient of the trajectories, and the safety of the robot (high oscillating velocities exalt the elasticity effect of tendons transmission).

It is worth noting that the commanded velocities are checked by a further lower level control: to prevent the robot to make fast movements, joint velocities $\dot{q}^*(t)$ are saturated in the range $[-25, +25](deg/s)$. This rough solution is necessary to avoid the stress of the elastic parts of the robot (mainly tendons) and their consequent damage. In this specific case, the regulator (9) assures asymptotic stability with faster response of the system with respect to the purely proportional regulator. The commanded Cartesian velocities are then:

$$v(t) = \dot{x}^*(t) + K_e\tau_e e(t) + K_e \int e(t)dt. \tag{10}$$

where $e(t)$ is the one of (8). Also in this case, it is possible to guarantee stability and convergence, for $\tau_e > 0, K_e > 0$ and the trend of convergence depends on the two eigenvalues resulting from $s^2 + k_e\tau_e s + k_e = 0$ ($K_e = k_e I$).

In Section III a method for the optimal trajectory planning in the Cartesian space, i.e. finding $x^*(t), \dot{x}^*(t)$, was discussed. More precisely, it has been shown how a FH or RH neural controller could be exploited for optimal planning: a neural control law $\hat{\mu}^\circ$ was used to find the optimal trajectory in the Cartesian space, which satisfied some performance criterion. A scheme illustrating how the neural network copes with the CLIK controller is presented in Fig. 5.

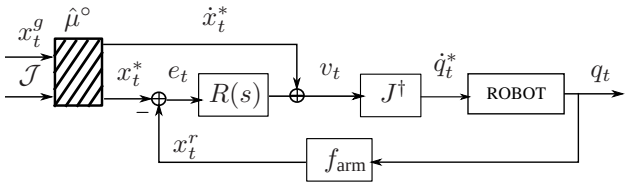

Fig. 5. James's arm CLIK controller, with the contribution, in evidence, of the neural controller.

V. EXPERIMENTAL RESULTS

In this experiment, a receding-horizon neural controller is used to plan Cartesian trajectories for reaching.

The desired trajectory is characterized by the following convex cost function: $\mathcal{J} = \sum_{i=t}^{t+N-1} c(u_i) + \xi_{i+1}^\top V_{i+1}\xi_{i+1}$ where $\xi = [\Delta x, \Delta\dot{x}, \Delta y, \Delta\dot{y}, \Delta z, \Delta\dot{z}]$, i.e. the difference between the target and the end-effector Cartesian positions and velocities, $u = [\ddot{x}, \ddot{y}, \ddot{z}] \triangleq [u^x, u^y, u^z]$. Note that $\mathcal{J}$ (in its general form, see Eq. 2) usually represents a tradeoff between the minimization of the energy consumption and the "best" end-effector proximity to the target during and at the end of the manoeuvre. The second term is indeed related to the distance to the target during the manoeuvre, which results in high velocity desired trajectories. Conversely, the first term acts as a damping quantity, which is necessary to reduce the risk of damage of the platform. Weight matrices V_i, in (**??**), were chosen such as to obtain reasonable compromise between the attractiveness of the target and the energy consumption. $c(u_i)$ is a nonlinear but convex function:

$$c(u_i^j) = \alpha \left[\frac{1}{\beta} \ln(2 + e^{\beta u_i^j} + e^{-\beta u_i^j}) - \frac{1}{\beta} \ln(4) \right], j = x, y, z \tag{11}$$

which, for large values of β approximates the ideal but non differentiable cost $\alpha \left| u_t^j \right|$, as shown in Fig. 6.

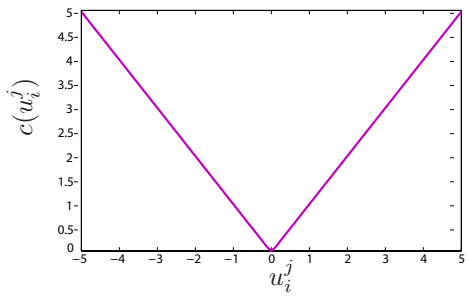

Fig. 6. The cost function (11), with $K = 0.01$ and $\beta = 50$.

In particular, $V_i = diag(1.0, 80.0, 1.0, 80.0, 5.0, 10.0)$, $i = 1, \ldots, N-1$, $V_N = 40I$. Output velocities were bounded within a safe range. Function f (see Eq. 1) is a double time integrator of the controlled accelerations u_i.

Remark 5: The double integrator approximates ideally the robot and the CLIK controller; the latter takes entirely into

account of singularities and redundancies, and is properly tuned to achieve the desired behaviors. Thus, the overall system performances do not degrade.

The training of the neural networks chain was performed off-line with the following parameters: $N = 60, \nu = 40$ (where N represents is the number of control steps, and ν is the number of total neurons of the net), $\varphi = \tanh$. More than 10^7 random samples were used to feed the networks, considering the whole reachable Cartesian space for the robot, and an augmented space for the target (to consider also unreachable targets); velocities and position were uniformly sampled. The neural networks' weights w and biases b were initialized with a slightly modified version of the known Nguyen-Widrow method [24], adapted to the multiple dimension case: $b = rand(0.7\nu^{(1/n)})$, $w = rand((0.7/n)\nu^{(1/n)})$ where ν, n are the number of neurons and the number of inputs of the neural network, respectively, and $rand(a)$ is a function extracting a random value within the range $[-a, a]$.

Remark 6: The flops count for the on-line computation of the approximated optimal controls is about 4633, which in a Pentium IV 3GHz are approximatively $10\mu s$.

In Fig. 7 the task is presented: the target is "fixed" (i.e. still), but changes unpredictably its position after a variable unknown period of time. This situation is representative of the case where the attentive system of the robot selects a target to be reached in the space (e.g. when the robot recognize a known object of interest). Cartesian trajectories of the end-effector and target position (named "desired") are shown in Fig. 8, while the corresponding arm's joint position and velocities are shown in Fig. 9. The saturation of velocities is evident in the joint velocity profiles.

Remark 7: Eq. 11 reminds the absolute work term [3], measuring the energy expenditure of a movement. In this case the goal is not to minimize an absolute work term (which would require the torques and thus a more complex dynamic model), however it is interesting to show that the proposed method can be applied to the implementation of "complex" nonlinear costs, which are usually superseded due to their mathematical difficulty (e.g. induced by the absolute value).

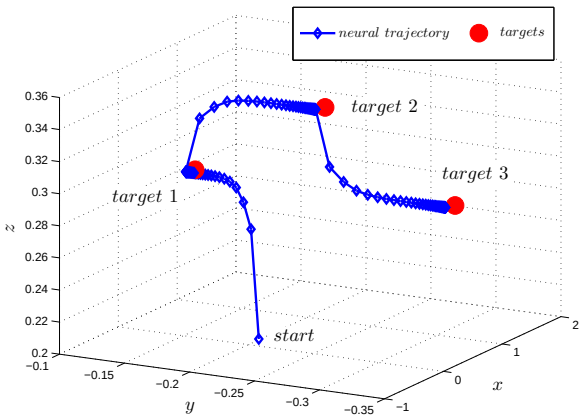

Fig. 7. "Neural" trajectory (blue) of the end-effector, reaching a target (red) which changes unpredictably. The shape of each trajectory is determined by the cost function $\mathcal{J}$ and its parameters.

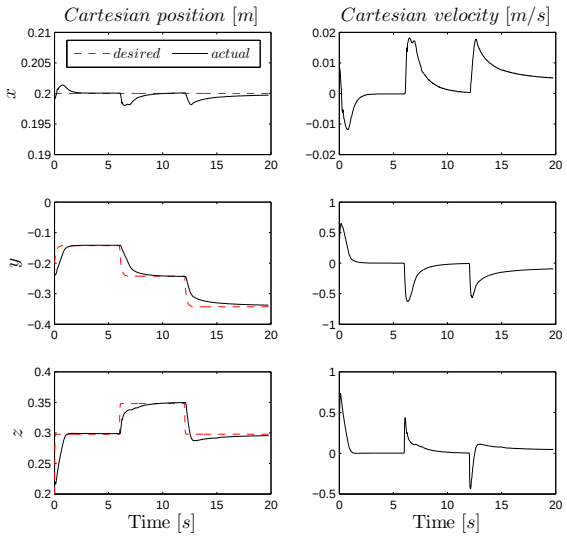

Fig. 8. Cartesian positions and velocities of the end-effector during the movements in Fig. 7. It is worth noting the different velocity profiles (e.g. $\dot{y}, \dot{z}$) due to the different parameters of the cost function.

VI. FUTURE WORK

Ongoing work deals with the control of the 7DOF arm (including orientation in the desired trajectory), and the effective implementation of the visual feedback. Remark 4 suggests the application of the method in presence of a non-linear function f, which could avoid the CLIK, solving the singularities and redundancy issues. Thus another problem will be addressed, as the estimation of such function is not trivial [31]. As pointed out in Remark 7, the proposed method can be applied with different cost functions, and given the generic formulation, in different contexts (i.e. for different state and control vector ξ, u). Lately James has been provided with a single force/torque sensor (FTS), placed in the middle of the arm [23]. Ongoing work is currently investigating how to exploit the proximal force sensing measurements to retrieve the joint forces and torques in the arm, with the purpose of implementing force/torque control. The future plan is to apply the proposed method for planning in the torque space, and in a mixed Cartesian/torque space, thus allowing the implementation of the absolute work principle. Furthermore, the FTS will provide the necessary information to detect contacts occurring during motion: this will enable to plan trajectories preserving safety during motion.

VII. CONCLUSION

In this paper on-line optimal trajectory planning is tackled through Finite and Receding Horizon Neural control. A reaching/tracking task for a 4DOF humanoid arm was presented, where the cost affecting the motion was particularly significant, showing the capability of the proposed method to handle nonlinear systems and/or nonlinear cost functions

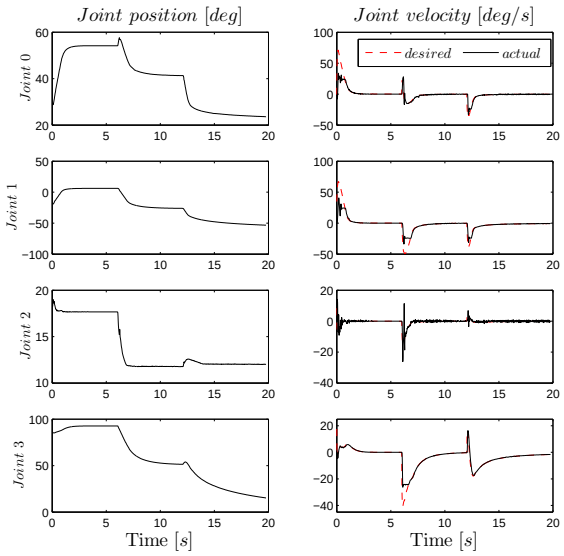

Fig. 9. Position and velocity of the arm's joints during the movements in Fig. 7.

without further complications. The trajectory generation and execution can be separated: this way the computational burden is concentrated in the off-line phase (controls are computed directly online without recurring to further optimizations), so that the computation of the (approximated) optimal controls can be performed quickly (almost instantaneously) and efficiently on-line, without incurring in real-time issues or being resource demanding.

VIII. ACKNOWLEDGMENTS

This work was supported by European Commission projects RobotCub (IST-004370) and CHRIS (ICT-215805).

REFERENCES

[1] E. Todorov and M. I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5:1226–1235, 2002.

[2] A. Biess, M. Nagurka and T. Flash. Simulating discrete and rhythmic multi-joint human arm movements by optimization of nonlinear performance indices. *Biological Cybernetics*, 95:31–53, 2006.

[3] B. Berret, C. Darlot, F. Jean, T. Pozzo, C. Papaxanthis and J.P. Gauthier. The Inactivation Principle: mathematical solutions minimizing the absolute work and biological implications for the planning of arm movements. *PLoS Computational Biology*, 4(10): e1000194, 2008.

[4] E. Guigon, P. Baraduc and M. Desmurget. Optimality, stochasticity and variability in motor behavior. *J. Computational Neuroscience*, 24(1):57–68, 2008.

[5] T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neuroscience*, 5:1688–1703, 1985.

[6] C.M. Harris and D.M. Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394:780-784, 1998.

[7] Y. Uno, M. Kawato and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement. *Biological Cybernetics*, 61:89-101, 1989.

[8] M.J. Richardson and T. Flash. On the Emulation of Natural Movements by Humanoid Robots. *Int. Conf. on Humanoids Robots*. USA, 2000.

[9] H. Seki and S. Tadakuma. Minimum jerk control of power assisting robot based on human arm behavior characteristics. *Proc. IEEE Int. Conf. on System, Man and Cybernetics*, 1:722-727, 2004.

[10] T.M. Tuan, P. Souères, M. Taïx and E. Guigon. A principled approach to biological motor control for generating humanoid robot reaching movements. *Proc. IEEE Int. Conf. Biomedical Robotics and Biomechatronics*, pp.783–788, 2008.

[11] O. Brock, J. Kuffner and J. Xiao. Motion for manipulation tasks. In *Siciliano, Khatib (Eds.): Springer Handbook of Robotics*. Springer, pp. 615–645, 2008.

[12] F. Nori, L. Natale, G. Sandini and G. Metta. Autonomous learning of 3D reaching in a humanoid robot *Proc. Int. Conf. On Intelligent Robots - IROS*, pp. 1142-1147, 2007.

[13] G. Simmons and Y. Demiris. Optimal robot arm control using the minimum variance model. *J. Robotic Systems*, 22(11):677–690, 2005.

[14] T. Matsui, M. Honda and N. Nakazawa. A new optimal control model for reproducing human arm's two-point reaching movements: a modified minimum torque change model. *Proc. IEEE Int. Conf. on Robotics and Biomimetics*, 1541–1546, 2006.

[15] T. Parisini and R. Zoppoli. A receding horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31:1443–1451, 1995.

[16] R. Zoppoli, M. Sanguineti and T. Parisini. Approximating networks and Extended Ritz Method for the solution of functional optimization problem. *Journ. of Optim. Theory and Applications*, 112:403-439, 2002.

[17] A. Barron. Universal approximation bounds for superposition of a sigmoidal function. *IEEE Trans. on Information Theory*, 39:930–945, 1993.

[18] S. Ivaldi, M. Baglietto and R. Zoppoli. Finite and receding horizon regulation of a space robot. *Proc. Int. Conf. on Mathem. Probl. in Engin. Aerospace and Sciences*, pp. 608-616, 2008.

[19] L. Sciavicco and B. Siciliano. Modeling and Control of Robot Manipulators, 2nd Edition. *Springer-Verlag*. London, UK, 2000.

[20] H.J. Kushner and J. Yang. Stochastic approximation with averaging and feedback: rapidly convergent "on-line" algorithms. *IEEE Trans. on Automatic Control*, 40:24–34,1995.

[21] V. Kurkova and M. Sanguineti. Error estimates for approximate optimization by the Extended Ritz method. *SIAM J. on Optimization*, 15:461-487, 2005.

[22] S. Ivaldi, M. Baglietto, G. Metta and R. Zoppoli. An application of receding-horizon neural control in humanoid robotics. *L. Magni et al. (Eds.): Nonlinear Model Predictive Control*. LNCIS 384, Springer-Verlag Berlin Heidelberg, pp. 541-550, 2009.

[23] M. Fumagalli, A. Gijsberts, S. Ivaldi, L. Jamone, G. Metta, L. Natale, F. Nori and G. Sandini. Learning to exploit proximal force sensing: a comparison approach. *O. Sigaud et al. (Eds.): From Motor Learning to Interaction Learning in Robots*, Studies in Computational Intelligence, vol. 264, pp. 159–177. Springer-Verlag, 2010.

[24] D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *Proc. International Joint Conference on Neural Networks*, 3:21-26, 1990.

[25] S. Chiaverini, O. Egeland and R.K. Kanestrom. Achieving user/defined accuracy with damped least/squares inverse kinematics. *5th Int. Conf. Advanced Robotics - ICAR*, pp. 672–677, Pisa, 1991.

[26] M. Diehl, H.G. Bock, H. Diedam and P.B. Wieber. Fast Direct Multiple Shooting algorithms for optimal robot control. *Fast Motions in Biomechanics and Robotics*. LNCIS 340, Springer Berlin / Heidelberg, pp. 65–93, 2006.

[27] M. Diehl, H.J. Ferreau and N. Haverbeke. Efficient numerical methods for nonlinear MPC and Moving Horizon estimation. *L. Magni et al. (Eds.): Nonlinear Model Predictive Control*. LNCIS 384, Springer-Verlag Berlin Heidelberg, pp. 541-550, 2009.

[28] A. Wächter and L.T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[29] S. Chiaverini, G. Oriolo and I.D. Walker. Kinematically redundant manipulators. In *Siciliano, Khatib (Eds.): Springer Handbook of Robotics*. Springer, pp. 245–268, 2008.

[30] L. Jamone, G. Metta, F. Nori and G. Sandini. James: a humanoid robot acting over an unstructured world. *Int. Conf. on Humanoids Robots*, pp. 143–150, 2006.

[31] E. Gribovskaya and A. Billard. Learning Nonlinear Multi-Variate Motion Dynamics for Real-Time Position and Orientation Control of Robotic Manipulators. *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*, pp.472–477, 2009.