Accurate Control of a Human-like Tendon-driven Neck

Francesco Nori, Lorenzo Jamone,
Giorgio Metta, Giulio Sandini
Department of Robotics, Behavior and Cognitive Sciences
Italian Institute of Technology
Genoa, Italy
Emails: francesco.nori@iit.it, lorenzo.jamone@iit.it,
giorgio.metta@iit.it, giulio.sandini@iit.it

***Abstract*— In this paper we describe the actuation and control of a humanoid robot neck. Particular attention will be posed on the description of the neck actuation structure, whose design has a noticeable human similarity. Specifically, the final mechanical design was inspired by the human skeleton, with the neck bone movements constrained and actuated by the surrounding muscles. In our robotic platform, the neck bone was realized with a steel spring surrounded by steel tendons in place of muscles. The specific and innovative mechanical design have imposed the design of a non-standard actuation structure which, in turn, have lead to an innovative control scheme. The main focus of the paper will be on describing different control schemes and discussing their performances in details.**

I. Introduction

Humans exhibit a wide and complex repertoire of movements far beyond the motor capabilities of modern robots. Clearly, the realization of an artificial system capable of more realistic movements passes trough a series of technological improvements, especially if we are interested in replicating both kinematic and dynamical aspects. Recently, there has been a growing interest in developing robots whose geometric and actuation structures resemble those of a human beings. Probably, the most extreme steps in this direction are represented by *Cronos* [1], the robot recently developed by O. Holland et al., and *Kotaro* [2], developed at Tokyo university. Along the same line, Albers et al. [3] have focused their attention on an innovative robotic neck, named *Vertebral Neck*, highly inspired by the human neck structure.

Noticeably, the control of these innovative architectures is a complex task, especially if compared with classical designs [4], [5] which have been usually based on rotational joints in serial configuration. Remarkably, the control complexity clearly increases with the architecture complexity and suggests new interesting control problems. Within this context Terzopoulos and Lee [6] have proposed an interesting solution to control an extremely detailed and precise biomechanical model of the human head-neck system. Though limited to a simulation environment, their work is to our knowledge one of the few considering the problem of controlling an highly realistic (muscle-actuated) model of the neck.

Realizing such complex architecture on a real humanoid robot is clearly an ambitious goal. Nevertheless, we believe in the importance of developing and implementing innovative and human-like actuation schemes within the field of humanoid robotics. Therefore, in this paper we aim at exploring the potentialities of a specific actuation structure: a spine joint actuated by the surrounding muscles. The proposed mechanical structure is not only simulated but also realized and controlled on a real humanoid robot neck. Clearly, the final solution makes major simplifications to reduce the whole system complexity but the mechanical structure preserves some interesting features which will be discussed in details.

The paper is organized as follows. In sections II and III we give a description of the hardware platform, focusing in particular on the neck structure. In section IV we explain in details the different control schemes we have tested. Finally, in section V we introduce future works concerning the development of a non-model-based controller for the neck and in section VI we present our conclusions.

II. Humanoid platform

The robotic platform on which the discussed controller has been implemented is the humanoid robot James [7]. James is a 22-DOF torso with moving eyes and neck, an arm and a highly anthropomorphic hand (see Figure 1). In the following subsections we briefly cover the robot design, its actuation, and sensorization. More details about the neck structure are given in section III.

A. Robot design

The robot structure is similar to that of humans, both in size (approximatively that of a ten-year-old boy), number of DOFs and range of movements; the total weight (about 8 kg: 2 kg the head, 4 kg the torso and 2 kg the arm and the hand together) has been kept low by the use of light material such as aluminum and ergal.

The head is equipped with two eyes, which can pan and tilt independently (4 DOFs), and is mounted on the 3-DOFs neck, which allows the movement of the head as needed in the 3D rotational space.

The arm has 7 DOFs: three of them are located in the shoulder, one in the elbow and three in the wrist. The hand

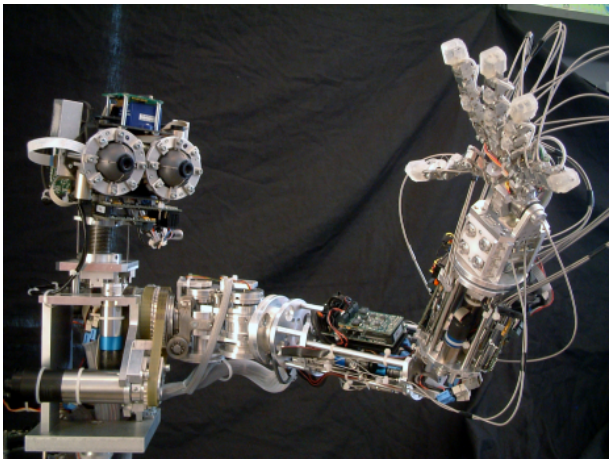

Fig. 1. The humanoid robot James.

has five fingers, with overall 17 joints coupled among them and actuated by just 8 motors.

B. Actuation system

The 22 DOFs are actuated by a total of 23 motors, whose torque is transmitted to the joints by plastic toothed belts and stainless-steel tendons, provided with springs at critical locations.
This solution is appealing for at least two reasons. First, it allows the housing of the motors far from the joints, thus distributing the weights in accordance with the designer wish. Second, the intrinsic elasticity of belts, tendons and springs gives a noteworthy compliance to the whole structure allowing the robot to move safely in a dynamic and unknown environment.

C. Sensory system

The robot is equipped with visual, proprioceptive, kinesthetic and tactile inputs. Vision is provided by two digital CCD cameras (PointGrey Dragonfly remote head), located in the eyeballs. The proprioceptive and kinesthetic senses are achieved through position sensors (magnetic incremental encoders connected to all motors); furthermore, a 3-axis orientation tracker (Intersense iCube2) has been mounted on top of the head, to emulate the vestibular system. Tactile information is extracted from several magnetic silicone-made pressure sensors which have been specifically designed and developed for James, placed in the fingers.

III. Neck structure

The neck bone is constituted by a steel spring, which holds the head giving it the possibility of bending forward (pitch) and laterally (roll). The actuation of these two degrees of freedom is obtained with a peculiar structure, recalling the design of a tendon-driven parallel manipulator; recent studies on this kind of actuation systems can be found in [8], [9]. Specifically, the neck is surrounded by three steel tendons, separated 120 deg apart. The tendons length determines the position of the spring and therefore, the pitch and roll position

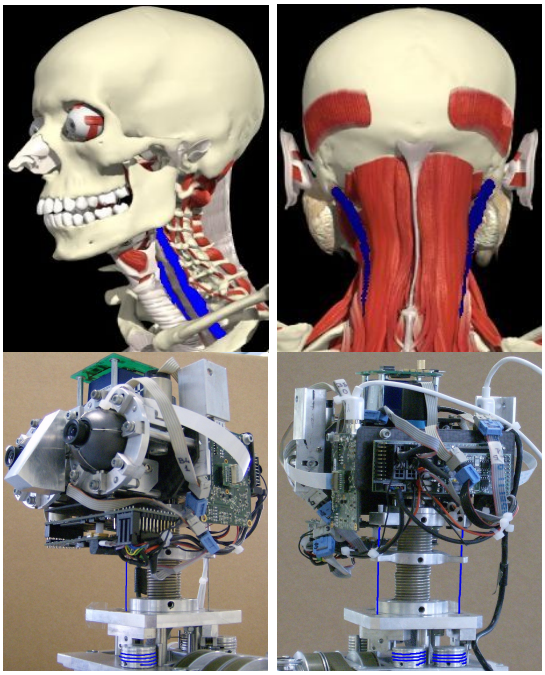

Fig. 2. Top. Human neck muscles we took inspiration from in the system design, highlighted in blue: *Longus Colli* in the left image, and *Longissimus Capitis* in the right image (images taken from *Primal 3D Anatomy* software [11]). Bottom. James neck tendons are arranged like the human muscles highlighted in the top images.

of the neck. The length of the tendons is adjusted by means of three motors, positioned at the base of the neck (see Figure 3).
This special geometry allows the neck to show ranges of motion comparable to the human ones; as claimed by Clarkson [10], average ranges of motion for pitch and roll rotations in adults are around ± 45 deg, while James motion has been bounded by software in the range of ± 40 deg (safe limit, lower than hardware limit).

Furthermore, this peculiar structure, even if far from being a close reproduction of the human musculoskeletal system, presents some analogies with the arrangement of some human neck muscles (see Figure 2). Humans are provided with more than 20 types of muscles in the neck, with several units for each type, settled into different layers: their work is both to control the orientation of the head respect to the neck base and to give stability to the cervical spine in supporting the weight of the head [11]. In our system, the spring tone is sufficient to support the head, and the tendons are employed just to move the head along its roll and pitch axes.
If we consider the deeper muscular layer around the neck, we can find a couple of long anterior muscles (*Longus Colli*, see top left image in Figure 2) that are responsible for head flexion, and a couple of long posterior muscles (*Longissimus Capitis*, see top right image in Figure 2) that are responsible for head extension. Both muscles are also involved in the lateral flexion of the neck, even if this movement is mostly actuated by other muscles (*Scalene Muscles*). These muscles completely

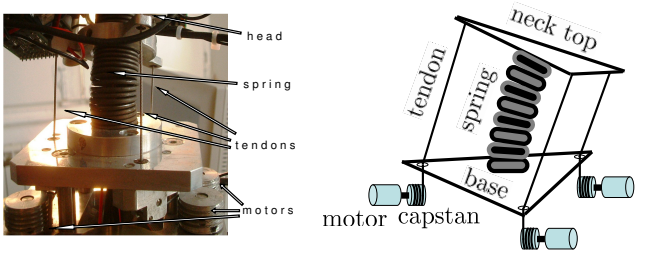

Fig. 3. Neck actuation system and sketch of the neck kinematics. Each motor pulls a tendon which passes trough a hole in the neck base. In this way the effective tendon length can be reduced to bend the spring in different directions.

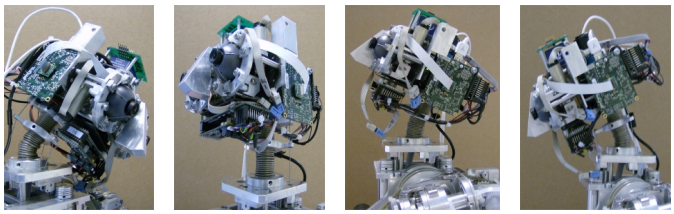

Fig. 4. James head. Different configurations seen from different views. Roll movements: left picture. Pitch movements: right pictures.

surrounds the cervical spine, having their origins, roughly speaking, at the level of the scapula, and their insertions at the level of the atlas; the anterior tendon in our system (bottom left image in Figure 2) can work as the *Longus Colli*, while the two tendons in the back (bottom right image in Figure 2) can emulate the *Longissimus Capitis*.
On top of the spring, a fourth independent motor is mounted, directly actuating a third degree of freedom, the head yaw (i.e. rotation around an axis parallel to the pan axes of the two eyes). In the current paper we are mainly interested in the control of the head pitch and roll by coordinating the movements of surrounding tendons.

A. Redundancy of the actuation scheme

Generally speaking, it is well known that a tendon-driven system with open-ended tendons (i.e. they can exert tension but not compression) requires more tendons than DOFs to be fully controllable. Therefore, to independently control n DOFs, $n+1$ tendons (and motors pulling the tendons) are needed [12]. Anyway, the peculiar parallel architecture of our system, makes it somehow redundant. In a mathematical sense, redundancy corresponds to the fact that the same configuration of the system can be achieved by different positions of the actuators. Classical techniques can be used to exploit the advantages of redundant systems [13]. However, in our case there are additional constraints that will guide the controller design in handling redundancy.

To understand the structure of the problem, let us reduce the structure to a two dimensional space. In this situation, we have two independent motors to actuate a single degree of freedom, nominally the slope of the surface on which the head is mounted. Consider first the system whose actuation scheme is given in Figure 5. With both motors we could actually control

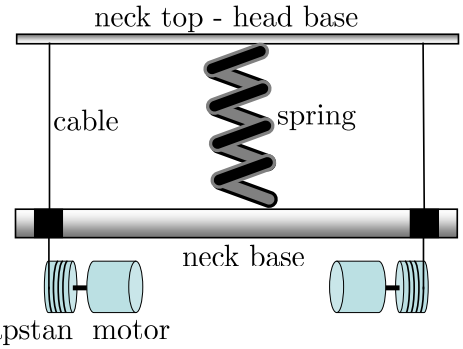

Fig. 5. Two dimensional scheme of an actuation system similar to James's neck.

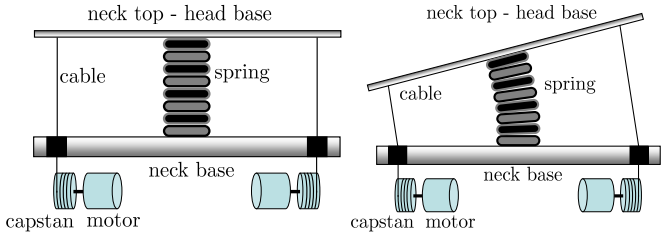

Fig. 6. Equivalent two dimensional scheme of james's neck.

two DOFs: the slope of the neck top (with respect to the neck base) and the spring compression. Therefore, considering the top surface slope as our control task, this system becomes redundant. Roughly speaking, shortening both cables of the same length does not produce any slope movement but only a variation of the spring compression. Classically, this is what we define redundancy in the actuation[1]. As previously said, in our case there are additional constraints that rule out this redundancy. In fact, the spring at the base of the neck is entirely compressed by the weight of the carried electronics and mechanics[2] (motors, cameras, chassis, etc. etc.), as indicated in Figure 6. As a consequence of this fact, shortening/lengthening both cables simultaneously no longer produces a variation of the spring compression but only varies the cables tensions. Remarkably, these tensions should be kept under control: high tensions damage (misalign) the spring spirals and low tensions cause wrong alignment of the cables on the capstans. Ideally, the tension of the cables can be controlled if we could use some sort of tension sensors. In our system these sensors are not currently available. Therefore, in the present paper we show how to keep the cable tensions under control by means of a kinematic model of the system.

[1]Note that, due to the presence of the spring, the neck top slope could be controlled with just one motor, even if with a limited workspace.

[2]The description of the system in Figure 5 is merely for understanding the issue of redundancy. Practically speaking it would be very difficult to model the forward kinematics of this system. The actual system, i.e. the one schematically represented in Figure 6, will be much easier to model kinematically. This is the reason why we did not choose a stiffer spring capable of sustaining the head weight.

IV. CONTROL OF THE NECK

The peculiar structure of the neck has required the design of an original control technique. The final design makes use of the 3-axis orientation tracker positioned on top of the robot head. Though the given sensor is capable of measuring its absolute rotation along three orthogonal axes, in the given application we used only part of the available information. In particular, we used the measurements corresponding to the head pitch and roll rotations, denoted θ_p and θ_r respectively; these rotations correspond to the degrees of freedom actuated by the three motors at the base of the neck. The third sensor measurement, the yaw rotation θ_y (a rotation around an axis orthogonal to the "neck top" as indicated in Figure 3), will not be used for two reasons: first, it is not influenced by the first three motor positions; second, it can be better measured using the encoder mounted directly on the shaft of the fourth motor.

A. Neck control in details

As already pointed out, the neck structure is characterized by three degrees of freedom: pitch θ_p, roll θ_r and yaw θ_y. The yaw movement, is directly actuated by a single dc motor; its control is based on a standard PID controller. The control strategy for the remaining two movements will be instead described in details in this section.

The design of the pitch and roll control loops has required the development of a neck structure model [3]. As already pointed out the system is somehow redundant (3 actuators versus 2 degrees of freedom) but the redundancy needs to be ruled out in order to keep the tendons tension within certain limits and the spring spirals aligned. Practically, because of redundancy, the same neck orientation $\mathbf{x} = [\theta_p, \theta_r] \in \mathbb{R}^2$ (i.e. the orientation tracker measurements) can be achieved with different tendons (cables) configurations $\mathbf{q} = [d_1, d_2, d_3] \in \mathbb{R}^3$ (i.e. the position of the motors). Mathematically speaking, the neck forward kinematic is described by a function:

$$\mathbf{x} = f(\mathbf{q}) \qquad f : \mathbb{R}^3 \rightarrow \mathbb{R}^2. \tag{IV.1}$$

The function f cannot be directly inverted because the same $\mathbf{x}$ can be achieved by different configurations $\mathbf{q}$. Among all these configurations, there is an ideal one $\mathbf{q}^*$ which corresponds to straight tendons and constant curvature of the spring. This configuration can be easily computed (see IV-B for details) thus leading to the following:

$$\mathbf{q}^* = f_{inv}(\mathbf{x}). \tag{IV.2}$$

The other tendons configuration corresponding to the same $\mathbf{x}$ are less desirable because of the reasons explained before. The reader should notice that (IV.2) is a sort of inverse kinematic model[4] expressing the configuration of the motors to achieve a desired neck orientation. Geometrically speaking, (IV.2) defines a two dimensional manifold $\mathcal{M}$ embedded in the three dimensional space of the cables configurations. Clearly, keeping the cable configuration $\mathbf{q}$ as close as possible to $\mathcal{M}$ is desirable because it corresponds to almost straight tendons (thus guaranteeing a minimum tendon tension) and almost constant curvature of the spring (thus guaranteeing aligned spring spirals). Therefore redundancy is ruled out by the additional requirement of remaining as close as possible to $\mathcal{M}$.

In the remaining of the current section we first give details about the kinematic model that leads to (IV.2), then we describe how to control the three motors in order to position the neck on a desired configuration $\mathbf{x}_d = [\theta_p^d, \theta_r^d]$ while keeping the position of the motors as close as possible to the manifold described by the kinematic model.

B. Partial inverse kinematic model

The three main assumptions are the following: the spring length is constant (remember that the spring is completely compressed by the head weight), it has a constant curvature and it always lays on a plane. Only the two extremities of the spring are attached to the robot, one to the fixed base of the neck (reference frame Σ_b) and the other to a movable plane on which the orientation tracker (reference frame Σ_s) is mounted (see also Figure 3). Practically the sensor measures the orientation of the this plane with respect to the gravitational force vector. In the remaining of this section, we express the sensor measurement in terms of the $\mathbf{z}$-axis of the reference frame Σ_s. Given the configuration of the system, this axis, denoted $\mathbf{z}_s$, is always parallel to gravity and can be easily expressed in terms of θ_p and θ_r:

$$\mathbf{z}_s = \begin{bmatrix} -\sin(\theta_r) & \sin(\theta_p)\cos(\theta_r) & -\cos(\theta_p)\cos(\theta_r) \end{bmatrix}^\top .$$

In order to compute the function f_{inv} which expresses the ideal tendons length $\mathbf{q}$ given the sensor measurement $\mathbf{x}$, we proceed by computing the position of Σ_s as a function of $\mathbf{x}$. Remarkably, notice that the orientation of Σ_s is already known given that $\mathbf{z}_s$ is known and the system does not rotate with respect $\mathbf{z}_s$. Therefore we are left with determining the translation of Σ_s, or equivalently the position of its origin O_s with respect to Σ_b. The first problem to solve consists in finding the plane $\mathcal{P}$ on which the spring lays. Given the assumptions, $\mathcal{P}$ is orthogonal to $\mathbf{z}_b$ and $\mathbf{z}_s$ and passes trough the origin O_b of Σ_b. These considerations easily follow from the fact that the spring extremities are always orthogonal to the planes on which they are attached. Therefore, $\mathcal{P}$ is uniquely determined by its normal $\mathbf{z}_n = \mathbf{z}_s \times \mathbf{z}_b$ and one of its points O_b. Once $\mathcal{P}$ has been uniquely determined we only need to determine the position of O_s within this plane. This is a two dimensional problem and can be easily illustrated (Figure 7).

[3]The model is based on the assumption that the spring has a constant length. Practically, when the spring bends on a side, it maintains its length on that side (remember that the spring is compressed by the head weight) while stretching on the opposite side. When the spring is bent, the assumption is that its curvature is constant along the entire spring length.

[4]In absence of modeling errors we have $f(f_{inv}(\mathbf{x})) = \mathbf{x}$, $\forall \mathbf{x} \in \mathbb{R}^2$. Notice that the forward counterpart (IV.1) is complicated to be analytically computed and its computation falls outside the scope of this paper.

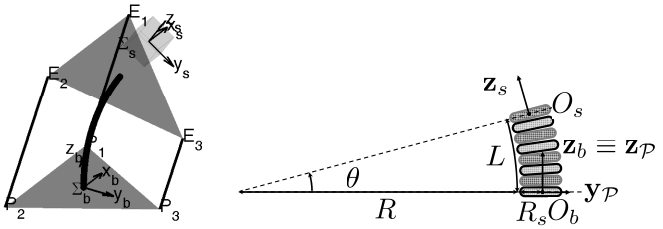

Fig. 7. Scheme for computing the inverse kinematic function f_{inv}. On the bottom picture, everything has been reduced to plane $\mathcal{P}$ to which both $\mathbf{z}_s$ and $\mathbf{z}_b$ belong.

We now restrict to the 2D case since the extension to 3D is just a matter of a rotation. Practically speaking, the spring describes an arc of a circle (see Figure 7) whose length is L (the length of the compressed spring) and whose angular amplitude is $\theta = \arccos(\mathbf{z}_b \cdot \mathbf{z}_b)$. The radius of this circle is therefore:

$$R = \frac{L}{\theta}.$$

Considering a two dimensional reference frame $(\mathbf{y}_\mathcal{P}, \mathbf{z}_\mathcal{P})$ with origin in O_b, the relative position between O_b and O_s is described by the vector $[(R+R_s)(1-\cos(\theta), (R+R_s)\sin(\theta)]$ being R_s the spring radius. This vector can be used to describe the position of O_s with respect to Σ_b and therefore the relative position between Σ_b and Σ_s. Once this position is known we can compute the cables length $\mathbf{q}$ given the system kinematic parameters; in particular we have $\mathbf{q} = \begin{bmatrix}|E_1 - P_1|, & |E_2 - P_2|, & |E_3 - P_3|\end{bmatrix}$ (see Figure 7) where E_i are all fixed in Σ_s and P_i are all fixed in Σ_b.

C. Control solutions

1) Purely model based solution: If the model were perfectly corresponding to the real system, the problem of orienting the neck in a desired configuration $\mathbf{x}_d$ would be easily solved by computing the desired tendons length $\mathbf{q}_d = f_{inv}(\mathbf{x}_d)$ and controlling the positions of the three motors[5] so as to regulate the tendons to the desired configuration. Practically speaking, every model has its own errors and therefore the proposed scheme will never orient precisely the neck. In order to check the quality of the model we evaluated the error in positioning the neck in the configuration $\mathbf{x}_d$. Specifically, we computed $\mathbf{e} = \mathbf{x} - \mathbf{x}_d$ where $\mathbf{x}$ is the sensor measurement after having positioned the motors in the configuration $\mathbf{q}_d^* = f_{inv}(\mathbf{x}_d)$ with the following control law:

$$\dot{\mathbf{q}} = -K_p(\mathbf{q} - \mathbf{q}_d^*), \tag{IV.3}$$

where $K_p \in \mathbb{R}^{3\times3}$ is an arbitrarily chosen gain matrix. It is well known that using (IV.3) is such that $\mathbf{q}$ converges to $\mathbf{q}_d^* \in \mathcal{M}$. Theoretically[6], in absence of modeling errors we should have $\mathbf{e} = 0$; practically, we obtained the errors in Figure 8.

[5]The three motors are equipped with encoders so that the motor position control has been easily achieved with a simple PID controller based on feedback from encoders.

[6]According to what we have seen in the previous section the f_{inv} have the following property $\mathbf{x} = f(f_{inv}(\mathbf{x}_d)) = \mathbf{x}_d$.

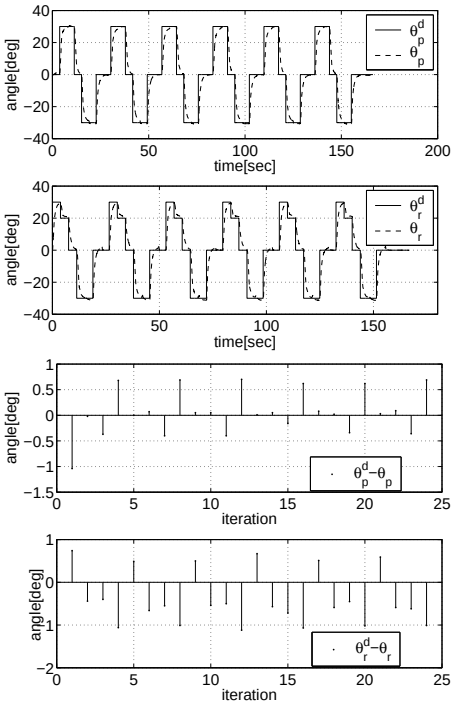

Fig. 8. Evaluation of the model. The bottom pictures show the errors $\mathbf{e} = \mathbf{x}_d - \mathbf{x} = [\theta_p^d - \theta_p, \theta_r^d - \theta_r]$ in performing a sequence of movements $\mathbf{x}_d = [30, 0] \rightarrow [20, 30] \rightarrow [0, 30] \rightarrow [-30, 0] \rightarrow [-30, -30] \rightarrow [0, -30]$. The sequence is repeated four times as shown in the top picture. The observed root mean squared error (RMSE) is $0.43\,\text{deg}$ for the pitch and $0.72\,\text{deg}$ for the roll.

Remarkably, the final configuration satisfies the requirements that we imposed on $\mathcal{M}$: constant spring curvature and straight tendons.

2) Jacobian based solution: In order to improve the positioning errors shown in Figure 8 we need to use the sensor measurement $\mathbf{x}$ as a feedback signal to reach the desired pose $\mathbf{x}_d$. A typical control structure for guaranteeing $\mathbf{x} \rightarrow \mathbf{x}_d$ is the following (see [13] for details):

$$\dot{\mathbf{q}} = -J_R(\mathbf{q})(\mathbf{x} - \mathbf{x}_d), \tag{IV.4}$$

where $J(\mathbf{q}) \in \mathbb{R}^{2\times3}$ is the Jacobian of (IV.1), and $A_R \in \mathbb{R}^{n\times m}$ denotes a right inverse of a matrix $A \in \mathbb{R}^{m\times n}$, i.e. $AA_R = I$. In our case, we do not have direct access to J since we do not have an analytical expression for the function f in (IV.1). All we have is f_{inv} whose Jacobian J_{inv} is somehow related to J. Specifically, it can be shown that:

$$J(f_{inv}(\mathbf{x}))J_{inv}(\mathbf{x}) = I \qquad \forall \mathbf{x} \in \mathbb{R}^2, \tag{IV.5}$$

which implies that J_{inv} is a right inverse for J in every $\mathbf{q}$ belonging to the manifold $\mathcal{M}$ described by (IV.2). Therefore, if we remain on that manifold, we can use J_{inv} in place of J_R in Eq. (IV.4). The final control law will be the following:

$$\dot{\mathbf{q}} = -J_{inv}(\mathbf{x})(\mathbf{x} - \mathbf{x}_d). \tag{IV.6}$$

Remarkably, the time derivative $\dot{\mathbf{q}}$ belongs to the tangent plane of $\mathcal{M}$ in $\mathbf{q}$. As a consequence of this fact, if the $\mathbf{q}(0) \in \mathcal{M}$

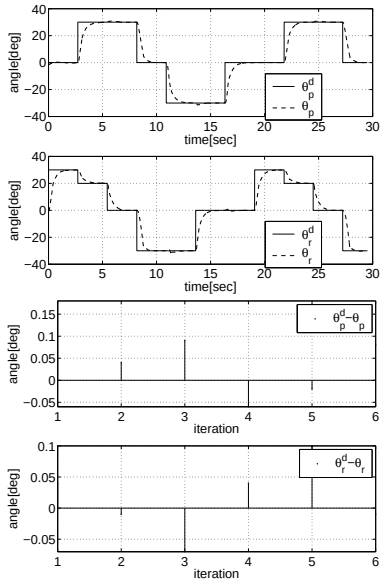

Fig. 9. Evaluation of the closed loop controller. The bottom pictures show the errors $\mathbf{e} = \mathbf{x}_d - \mathbf{x} = [\theta_p^d - \theta_p, \theta_r^d - \theta_r]$ in performing a sequence of movements $\mathbf{x}_d = [30, 0] \rightarrow [20, 30] \rightarrow [0, 30] \rightarrow [-30, 0] \rightarrow [-30, -30] \rightarrow [0, -30]$. The actual movement is shown in the top pictures. The observed root mean squared error (RMSE) is $0.09\,\mathrm{deg}$ for the pitch and $0.05\,\mathrm{deg}$ for the roll.

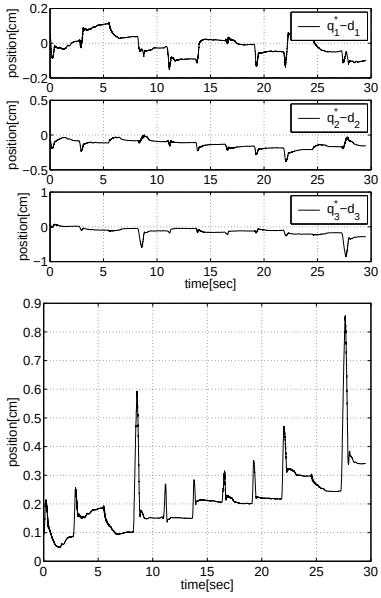

Fig. 10. Distance of $\mathbf{q}$ from the manifold $\mathcal{M}$. The top picture shows the three components of the vector $f_{inv}(\mathbf{x}) - \mathbf{q} = \mathbf{q}^* - \mathbf{q}$ during the movement described in Figure 9. The bottom picture shows instead the time behaviour of its norm , i.e. $\|\mathbf{q}^* - \mathbf{q}\|$. As expected the system configuration progressively abandons the manifold.

then $\mathbf{q}(t) \in \mathcal{M}, \forall t > 0$. Therefore, (IV.6) is exactly equivalent to (IV.4) and convergence of $\mathbf{x}$ to $\mathbf{x}_d$ is guaranteed. However, from a practical point of view, the implementation of (IV.6) is obtained with a digital controller so that the commanded velocity $\dot{\mathbf{q}}$ is not continuously modified but only updated every $T_s = 0.01$ seconds, the controller rate. Therefore, we are not guaranteed that the system configuration remains on $\mathcal{M}$. Figure 9 shows the performance of the discrete time controller. Positioning errors are greatly improved with respect to the previously proposed controller (IV.3); the pitch RMSE has been reduced from $0.43\,\mathrm{deg}$ to $0.09\,\mathrm{deg}$ while the roll RMSE from $0.72\,\mathrm{deg}$ to $0.05\,\mathrm{deg}$. However, as a consequence of the discretization of the controller, the system is not guaranteed to remain on $\mathcal{M}$ as shown in Figure 10.

3) Weighted solution: Keeping the system configuration on the manifold $\mathcal{M}$ is important for the reasons illustrated in Section IV-A. Practically, running (IV.6) for a long time results in misalignment of the spring spirals (high tension on one of the cables) and/or wrong alignment of the cables on the motor capstans (low tension). Therefore the solution proposed in the previous section is not desirable. According to Figure 10 the longer we control the system the bigger becomes the distance from manifold. The first solution to this problem is a mixture of the two proposed controllers (IV.3) and (IV.6):

$$\dot{\mathbf{q}} = -\mu K_p(\mathbf{q} - f_{inv}(\mathbf{x}_d)) - (1-\mu)J_{inv}(\mathbf{x})(\mathbf{x} - \mathbf{x}_d), \quad \text{(IV.7)}$$

where $\mu \in [0, 1]$ is an arbitrary weighting scalar factor and the other quantities have been defined in the previous sections. Choosing the value of the variable μ is not an easy task since it is meant to privilege either the distance to the manifold ($\mu \simeq 1$) or the achievement of a precise positioning ($\mu \simeq 0$). Choosing $\mu \simeq 0.5$ leads to the results in Figure 11. Remarkably the distance from the manifold is reduced (the observed RMSE is $0.1cm$) at the cost of an increased error in the positioning: the pitch-RMSE drops down from $0.09\,\mathrm{deg}$ to $1.32\,\mathrm{deg}$, while the roll-RMSE from $0.05\,\mathrm{deg}$ to $1.17\,\mathrm{deg}$.

4) Minimization solution: The control law (IV.7) is always a compromise between accuracy in positioning and accuracy in remaining close to the manifold $\mathcal{M}$. Practically, we cannot achieve both tasks simultaneously because the definition of $\mathcal{M}$ is based on a model of the system which is usually affected by modeling errors. The best way to satisfy both tasks simultaneously consists in requiring accurate positioning ($\mathbf{x} = \mathbf{x}_d$) while requiring the system configuration $\mathbf{q}$ to be as close as possible to the manifold configuration $\mathbf{q}^* = f_{inv}(\mathbf{x})$. Practically, to reach the orientation $\mathbf{x}_d$ we should move the system to the configuration:

$$\mathbf{q}_d = \min_{\mathbf{q}} \frac{1}{2}\|\mathbf{q} - f_{inv}(\mathbf{x}_d)\|^2, \qquad f(\mathbf{q}) = \mathbf{x}_d. \quad \text{(IV.8)}$$

The so called resolved motion rate control technique (see [13] for details and proof of convergence) can be used to solve the above problem:

$$\dot{\mathbf{q}} = -J_R(\mathbf{q})(\mathbf{x} - \mathbf{x}_d) - \left(I - J_R(\mathbf{q})J_R^{\dagger}(\mathbf{q})\right)(\mathbf{q} - f_{inv}(\mathbf{x}_d)), \quad \text{(IV.9)}$$

where J_R is a right inverse of the Jacobian $J = \frac{\partial f}{\partial \mathbf{q}}$ and $J_R^{\dagger}$ is its Moore-Penrose pseudo-inverse. This control strategy can be proved to guarantee that $\mathbf{q}$ converges to the solution $\mathbf{q}_d$ of (IV.8). Once more, an implementation of (IV.9) on our system is complicated by the fact that we have access neither to the

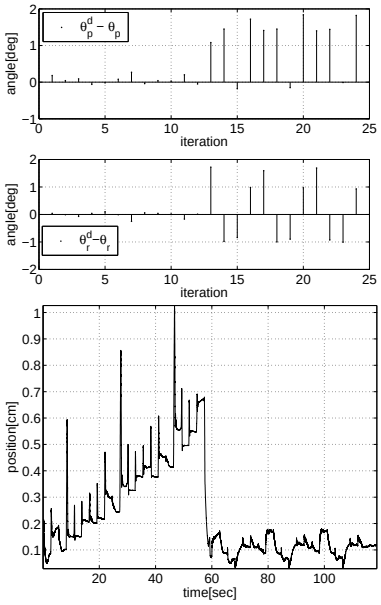

Fig. 11. Top: positioning errors $\mathbf{x}_d - \mathbf{x}$ for the two components of $\mathbf{x} = [\theta_p, \theta_r]$; bottom: distance of $\mathbf{q}$ from the manifold $\mathcal{M}$. The usual sequence of movements has been repeated twice with the controller (IV.4) and twice with (IV.7). The switching between the two controllers happens after about 60 seconds causing a rapid fall of the distance from the manifold.

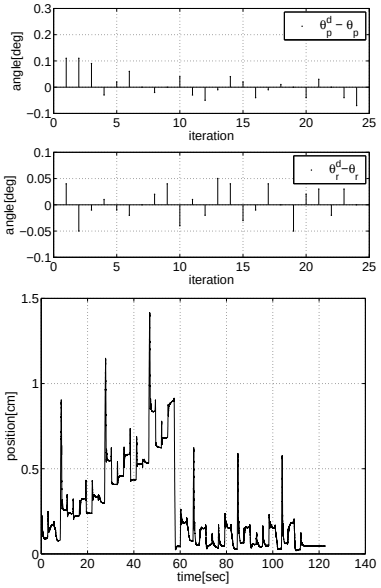

Fig. 12. Top: positioning errors $\mathbf{x}_d - \mathbf{x}$ for the two components of $\mathbf{x} = [\theta_p, \theta_r]$; bottom: distance of $\mathbf{q}$ from the manifold $\mathcal{M}$. The usual sequence of movements has been repeated twice with the controller (IV.4) and twice with (IV.10). The switching between the two controllers happens after about 60 seconds causing a rapid fall of the distance from the manifold.

function f nor to its Jacobian J. Using the same idea proposed before we have that as long as we stay on the manifold the following relationship holds: $J_R = J_{inv}$. Therefore, we can implement (IV.9) as follows:

$$\dot{\mathbf{q}} = -J_{inv}(\mathbf{x})(\mathbf{x}-\mathbf{x}_d) - \Big(I - J_{inv}(\mathbf{x})J^{\dagger}_{inv}(\mathbf{x})\Big)\left(\mathbf{q} - f_{inv}(\mathbf{x}_d)\right). \tag{IV.10}$$

Clearly, (IV.10) is similar to (IV.7) but in this case the manifold distance is controlled by performing movements in the null space of the primary task, the neck orientation. Theoretically, (IV.9) will be equal to (IV.10) only if we guarantee to remain on the manifold. Practically, all we are interested in is the convergence of the control to the desired final configuration; fairly weak conditions to guarantee the convergence can be found in [13]. These conditions[7] can be used to prove convergence of the proposed control scheme under the assumption that the system configuration remains close enough to $\mathcal{M}$. Further and more formal investigations on this sketch of convergence proof will be given on a forthcoming paper. The results concerning this control strategy compared with the previously proposed controller (IV.4) are shown in Figure 12. Remarkably the distance from the manifold is reduced (the observed RMSE is $0.3cm$) while maintaining a good positioning accuracy: the pitch-RMSE is $0.03\,\mathrm{deg}$, while the roll-RMSE is $0.03\,\mathrm{deg}$.

V. Future work

Preliminary works have been carried out concerning the development of a non-model-based controller for James neck, inspired by the recent attempts to use machine learning techniques to control robotic devices without having any model of the system in consideration (*Black Box* systems). Some examples can be found in [14]–[17].

The planned solution makes use of a Receptive Fields Neural Network (RFNN) to estimate the joint-space velocities needed to move the head with the desired task-space velocities, given the actual joint configuration. The learning algorithm is an implementation, with some modifications, of the Receptive Fields Weighted Regression proposed by Schaal and Atkenson [18].

After a training phase, in which the workspace is randomly explored by the robot and the network learns on-line from the gathered sensory data, the trained network is able to map the two spaces (i.e. it has learnt an inverse of the Jacobian matrix). Unfortunately, as previously stated, due to the redundant actuation system we deal with, our Jacobian matrix is not square, and so not invertible. This means that for a given set of actual joints positions and required task-space velocities there is not a unique solution in terms of joint-space velocities; nevertheless, within this family of solutions, just a very limited set is usable in practice, because a minimum amount of tension is needed along the tendons to avoid their possible outgo from the capstans.

To overcome this problem, a solution similar to the one presented in IV-C.3 as been adopted. The velocities applied to the motors during the training phase, $\dot{\mathbf{q}} \in \mathbb{R}^3$, result from the weighted summation of two terms, as in IV.7, being the second one a conveniently bounded random value, $\dot{\mathbf{q}}_{rnd} \in \mathbb{R}^3$,

[7] In practice convergence is guaranteed if $JJ_{inv} > 0$ which is obviously true on the manifold $\mathcal{M}$ where $J_{inv} = J_R$ so that we have $JJ_{inv} = I$.

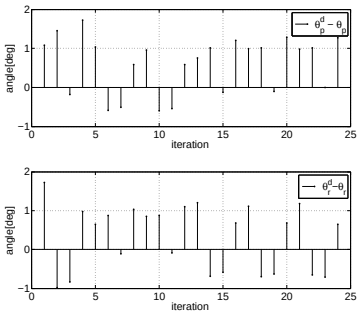

Fig. 13. Positioning errors $\mathbf{x}_d - \mathbf{x}$ for the two components of $\mathbf{x} = [\theta_p, \theta_r]$. The usual sequence of movements has been repeated four times with the RFNN controller. The measured RMSE is $0.92\,\mathrm{deg}$ for the pitch and $0.88\,\mathrm{deg}$ for the roll.

instead of $J_{inv}(\mathbf{x})(\mathbf{x} - \mathbf{x}_d)$:

$$\dot{\mathbf{q}} = -\mu K_p(\mathbf{q} - f_{inv}(\mathbf{x})) - (1-\mu)\dot{\mathbf{q}}_{rnd}. \qquad \text{(V.1)}$$

The data provided to the learning algorithm are the $3D$ vector of the applied joint-space velocities, $\dot{\mathbf{q}} \in \mathbb{R}^3$, as output, and the $5D$ vector composed by the resulting task-space velocities, $\dot{\mathbf{x}} \in \mathbb{R}^2$, and the actual motor configuration, $\mathbf{q} \in \mathbb{R}^3$, as input. Once the RFNN has learnt enough, we can use it to control the neck, giving the desired task-space velocities and the actual motor configuration as input, $[\dot{\mathbf{x}_d}, \mathbf{q}]$, and obtaining the appropriate joint-velocities as output, $\dot{\mathbf{q}}_d$.

In this way the redundancy is solved choosing the solution that keeps the tendons closer to their optimal lenght, exploiting the kinematic model in IV.2. Again the weight μ has to be tuned in order to balance the randomness of the exploration and the respect of the kinematic model (necessary to keep the tendons into the capstans grooves).

A first stage of development for the discussed controller has been already reached, and an initial version has been tested on James. Results show that the system is able to learn a good approximation of the inverse Jacobian after a training stage of about a hour, with pitch and roll RMSE lower than in the IV-C.3 solution, under $1\,\mathrm{deg}$ (see Figure 13).

VI. Conclusions

We proposed an innovative neck mechanical structure which is highly biologically inspired. The main idea is to have a flexible structure (neck bone) actuated by surrounding contractile elements (muscles). In the specific implementation the flexible structure is represented by a steel spring. Its actuation is achieved with standard dc motors which are used to pull three tendons surrounding the neck. The final structure is capable of bending the neck roughly around a sphere. This new mechanical design has required the development of an appropriate control structure which uses an orientation tracker, a model of the system kinematics and a Jacobian based feedback loop. The peculiarity of the system made it difficult to model directly its forward kinematics; practically, it was easier to model a specific solution of the inverse kinematics. On the basis of this solution, a suitable control strategy was proposed and implemented on the real robot. A formal mathematical proof of the convergence of the proposed control scheme was not given even if simple considerations seem to pave the way to a complete formal proof. In any case the controller has been practically shown to work in line with the required performance (positioning errors $< 0.05\,\mathrm{deg}$). Finally, hints about a non-model-based control scheme have been given.

References

[1] O. Holland and R. Knight, "The anthropomimetic principle," in *Symposium on Biologically Inspired Robots, Bristol, England*. AISB, April 2006.

[2] I. Mizuuchi, T. Yoshikai, Y. Sodeyama, Y. Nakanish, A. Miyadera, T. Yamamoto, T. Niemel, M. Hayashi, J. Urata, Y. Namiki, T. Nishino, and M. Inaba, "Development of musculoskeletal humanoid kotaro," in *International Conference on Robotics and Automation (ICRA)*. IEEE, May 2006.

[3] A. Albers, S. Brudniok, and W. Burger, "The mechanics of a humanoid," in *International Conference on Humanoid Robots*, Karlsruhe, Germany, 2003.

[4] H. Kim, G. York, G. Burton, E. Murphy-Chutorian, and J. Triesch, "Design of an anthropomorphic robot head for studying autonomous development and learning," in *International Conference on Robotics and Automation (ICRA)*. New Orleans, LA, USA: IEEE, April 26-May 1 2004.

[5] F. Guenter, L. Roos, A. Guignard, and A. Billard, "Design of a biomimetic upper body for the humanoid robot robota," in *International Conference on Humanoid Robots*. Tsukuba, Japan: IEEE, December 5-7 2005.

[6] S.H.Lee and D.Terzopoulos, "Heads up! biomechanical modeling and neuromuscular control of the neck," in *SIGGRAPH Conference, Boston, MA*. ACM, Aug 2006.

[7] L. Jamone, F. Nori, G. Metta, and G. Sandini, "James: A humanoid robot acting over an unstructured world," in *International Conference on Humanoid Robots*, Genova, Italy, 2006.

[8] R. Verhoeven and M. Hiller, "Estimating the controllable workspace of tendon-based stewart platforms," in *7th International Symposium on Advances in Robot Kinematics (ARK)*, Portoroz, Slovenia, 2000, p. 277284.

[9] A. Hay and J. Snyman, "Optimization of a planar tendon-driven parallel manipulator for a maximal dextrous workspace," vol. 37(3), pp. 217 – 236, April 2005.

[10] H. Clarkson, *Musculoskeletal Assessment*, 2nd ed. Lippincott Williams and Wilkins, 2000.

[11] F. Kendall, E. McCreary, P. Provance, M. Rodgers, and W. Romani, *Muscles: Testing and Function with Posture and Pain*, 5th ed. Lippincott Williams and Wilkins, 2005.

[12] L. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. Wiley-Interscience, 1999.

[13] C. Samson, B. Espiau, and M. L. Borgne, *Robot Control: the Task Function Approach*. Oxford University Press, 1991.

[14] Y. Zhang, J. Wang, and Y. Xia, "A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits," vol. 14(3), pp. 658 – 667, May 2003.

[15] S. Iida, K. Kuwayama, M. Kanoh, S. Kato, T. Kunitachi, and H. Itoh, "Humanoid robot control based on reinforcement learning," in *International Symposium on Micro/Nanomechatronics and Human Science (MHS)*, Nagoya, Japan, October 31- November 3, pp. 353–358.

[16] J. Conradt, G. Tevatia, S. Vijayakumar, and S. Schaal, "On-line learning for humanoid robot systems," in *17th International Conference on Machine Learning (ICML)*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 191–198.

[17] S. Mahadevan, "Machine learning for robots: A comparison of different paradigms," in *Workshop on Towards Real Autonomy , IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-96)*. Osaka, Japan: IEEE, 1996.

[18] S. Schaal and C. G. Atkenson, "Constructive incremental learning from only local information," *Neural Computation*, vol. 10(8), pp. 2047–2084, 1998.