International Journal of Applied Artifical Intelligence in Engineering System 1(1), 2009: 7-22

IJAAIES

EFFICIENT INVERSE KINEMATIC ALGORITHM FOR A TRULY ANTHROPOMORPHOUS ARTIFICIAL ARM

Giuseppina Gini*, Michele Folgheraiter* & Massimo Cavallari*

Abstract: This paper proposes a general methodology to solve the inverse kine- matic (IK) problem of an anthropomorphous artificial arm. We base our work on an arm moved by McKibben actuators. We formalized an hybrid efficient technique that combines the forward kinematic solution expressed in a close form with an optimization algorithm based on the gradient descent.

The advantage of using this approach is that is not necessary to explicitly solve the equations that link the position and orientation of the wrist with the vector that defines the actuators lengths. Furthermore we implemented an algorithm to adjust the robot precision in order to comply with the specific requirements on the task and the computational power available on board.

We tested the algorithms on the robot prototype and measured the precision and repeatability of the entire system in executing a target reaching behavior and a cyclic trajectory.

Keywords: Inverse Kinematic, Artificial Arm, Parallel Robotics, Hu-manoid Robotics

1. INTRODUCTION

The inverse Kinematic problem is a crucial issue for every manipulation system. A classical procedure to obtain the solutions requires first to calculate the forward kinematic using the well know conventions proposed by Denavit- Hartenberg [1] and Craig [2]. Then it is necessary to invert the equations finding the function that allows to calculate the joints positions vector θ given as input the position and orientation vector X of the end-effector $\theta = F^{-1}(X)$.

If the robot has a complex or redundant kinematics, it could be very difficult to find an analytical solution, and if the kinematic chain is redundant it is also necessary to discriminate one of the many available solutions. Another issue is due to the singularity points inside the workspace; in this case the Jacobian matrix cannot be inverted and consequently it brings some discontinuities into the joint workspace. This means that a continuous tra jectory in the Cartesian space has a discontinuous one in the joint workspace. When this happens the robot may change its posture very rapidly, representing a danger both for its integrity and the human operators near to it. If this is an issue for industrial manipulators, it is a big problem for a humanoid robot that is supposed to work in contact with the human beings during cooperative activities.

Furthermore in a robot arm with an anthropomorphous architecture open kinematics chains are combined with closed ones [3] [4], and this normally complicates the solution also for the direct kinematic problem. The main issue is that the solution of the direct kinematics for a parallel robot requires to verify if the combination for the joint positions is admissible, and this usually requires to solve a systems of non linear equations.

It becomes therefore necessary to adapt and combine classical techniques to reach a good tradeoff between precision real time performances [5] [6], and singularities avoiding.

Our approach to solve the inverse kinematics is based on the fast descent gradient algorithm in combination with an autoregressive method to choose the next step of descent. Furthermore to avoid the local minima and reach

* Department of Electronic and Information, Politecnico di Milano Piazza L. da Vinci 32, MILANO, I-20133, Italy.

the optimal solution we adopted the simulated annealing technique. In literature it is possible to find some works in this direction; in [7] the authors propose an hybrid technique that uses both the analytical solution and an optimization criterium to solve the inverse kinematics (IK) of a manipulator. The idea is to use the analytical solution for IK when it is possible and an optimization method, based on the descent-gradient, when the first methodology gives an unacceptable error. Authors underline that it is not possible to apply only the optimization method for the IK solution due to its computational inefficiency. In the work proposed by Stefan Schaal et al. [8] the IK problem for a humanoid robot is differently approached. The authors improved the performances of a technique that combines an optimization criterium with the inversion of the extended Jacobian matrix. This allows to elegantly solve the problem of singu- larities by adding new constrains to the system, and at the same time maintains the solution conservative and optimal.

Another advantage of using the gradient descent method is the possibility to integrate in a easy way the vision feedback in the control loop. This allows also to exclude the direct kinematic equation from the algorithm and to avoid the error in positioning the arm due to the sensors. Of course this will also bring new issues mainly due to the stereo-vision system performances.

2. THE HUMANOID ARM MAXIMUMONE

The methodology we proposed was applied to solve the inverse kinematic problem for the robotic arm we have developed in our laboratory. MaximumOne [9]was designed using the paradigm of the bio-robotics [10] [11] [12], the main idea being to use the knowledge on the anatomy and physiology of the human limb in order to synthesize a system with comparable performance from the kinematic point of view [13] [14] [15].

The system we consider is actuated by seven linear artificial muscles that control the four degrees of freedom of the kinematic chain. The actuators are connected to the links by flexible tendons that allow the actuator to change its orientation according to the links direction. We chose pneumatic McKibben actuators [16] for several reasons. They are efficient, since an actuator of 10 grams can develop a force of 200N; they are intrinsically elastic [17] so their use increases the safeness of the robot and facilitates the stiffness regulation. Furthermore they mimic the human muscle at least from the macroscopic point of view. Of course this kind of actuators brings also problems, since their dynamic behavior is not linear [18] [19] [20] and they are subject to hysteresis that com- plicates the control strategy and decreases the system precision. Anyhow our models and our control strategy are general and will remain valid also in future when more performing actuation systems could be available. Very promising are the ones based on electro-active polymers [21] that can be easily controlled modulating the current like DC motors.

The robot has a spherical joint in the shoulder that allows three DOF and a rotational joint in the elbow with one DOF. In particular the shoulder is moved by five artificial muscles:

- Deltoid (De)
- Pectoralis (Pe)
- Dorsal (Do)
- Supraspinatus (Sup)
- Subscapularis (Sub)

The synergy of the Deltoid, Pectoral and Dorsal actuators allows the upper arm flexion-extension and adduction-abduction. The Supraspinatus and Subscapularis instead act to rotate the upper Arm around its longitudinal axis. Thanks to a special design for the shoulder this third degree of freedom is kinematically independent from the first two. This is possible because of the rotation of the connection point between the two antagonist actuators and the joint (Figure 2 (a)). In this manner during the action of De-Pe-Do actuators the tendon that connects Sup and Sub can slide along the sphere without changing their lengths.

Due to the chosen actuation architecture for the shoulder joint, we can recognize a closed kinematic chain composed by the shoulder joint, the upper arm link, and the three actuators that govern the first two rotational DOFs (see figure 2 (b)). In formalizing the kinematic model we considered the actuators always tensed, in order to simplify the model and to apply the known theory of parallel robotics. Using the 3D Gruebeer formula it is possible to calculate the overall mobility of this kinematic chain:

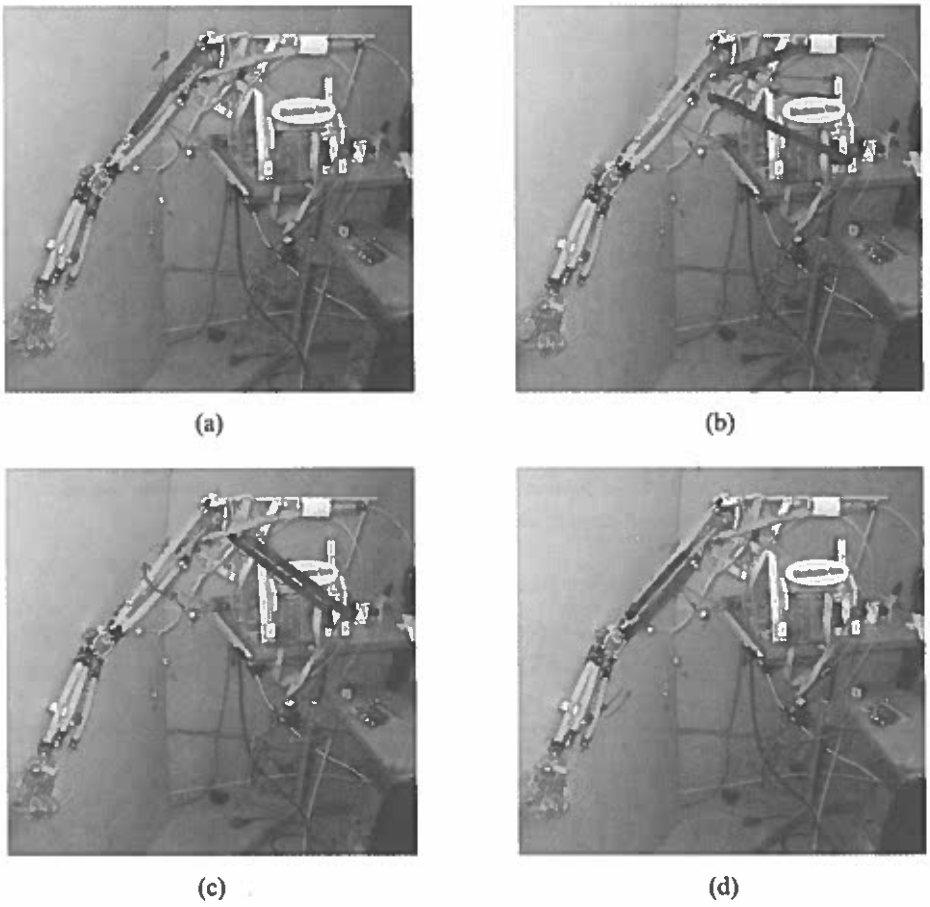

Figure 1: (a) The Deltoid Actuator Lifts the Shoulder. (b) The Pectoral and Dorsal Actuators Allow the Adduction and Abduction Movements. (C) The Supraspinatus and Subscapularis Actuators Allow the Shoulder Rotation (d) The Biceps and Triceps Actuators Allow the Elbow Flexion and Extension

$$m = 6(n - g - 1) - \sum_{g}^{i=1} f_i = 6(5 - 7 - 1) \_ 21 = 3 \tag{1}$$

where n is the number of links in the kinematic chain including also the ground connection (counted only once), g is the number of joints, and f_i is the number of DOFs for the ith joint. In the parallel chain we have six rotational joints with 3 DOF each, and 5 links considering the three actuators, the upper arm and the ground. As it is possible to note from equation 1 the kinematic chain that represents the shoulder has a total of three degrees of freedom one of which is actuated by the Sup. and Sub. actuators. We do not consider them in this model because they are mechanically independent from the others.

The Elbow joint is also actuated by two dedicated actuators:

- Biceps (Bi.)
- Triceps (Tri.)

Both Biceps and Triceps origin from a point of the upper arm and connect with the forearm, the first in the frontal side of the link and the other in the backside. This allows the movements of flexion and extension of the forearm.

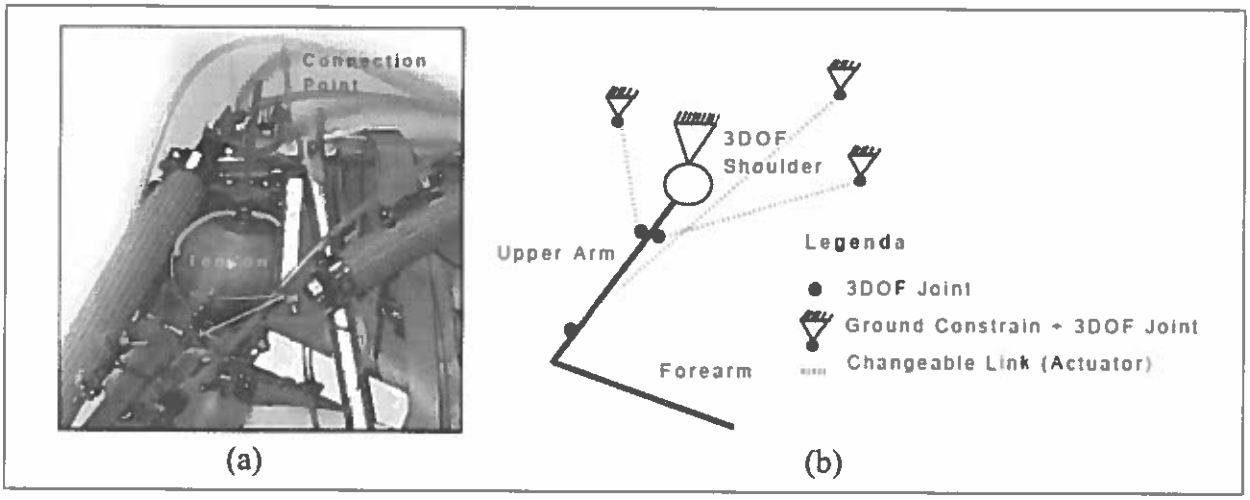

Figure 2: (a) A Special Pivot Separate the Third Rotational Degree of Freedom from the First Two (b) Schema of the Parallel Kinematic Chain Composed by the Shoulder Joint, the Upper Arm Link, and Three Artificial Muscles

The arm is equipped also with a five fingers hand, not considered in this study.

Because the arm prototype has only four degrees of freedom, without considering the wrist mobility, we are able only to fix the hand (wrist) position not its orientation. This does not affect the methodology proposed, at the condition that further constrains are introduced in the kinematic chain.

3. DIRECT KINEMATICS

Solving the direct kinematic problem for the anthropomorphic arm means to find the function that allows us to calculate the wrist position and orientation given as input the lengths vector for the actuators L.

If we discard, just for now, the wrist orientation it is possible to represent the wrist position P^0_{wrist} relative to the arm absolute reference system, given as an input the length vector L for the artificial muscles by equation 2.

$$P^0_{wrist} = F(L) \tag{2}$$

In equation 2 the length vector $L \in R^7$ and $F:(R^7 \rightarrow R^3)$ is a vectorial field. It is relevant to note that not all the seven muscles lengths are independent variables. This is immediately clear if we think, for example, that for a certain wrist position it is not possible to set any arbitrary length for both the biceps and triceps actuators. As we told before the arm has, without considering the wrist and hand mobility, only four DOF's; this means that only four of the seven variables (muscles lengths) are free.

In this case the equation 2 will assume the simplified form of equation 3, where only four of the seven actuators are considered. Furthermore with four variables we can basically fix an arbitrary position for the wrist within the robot workspace and still have another variable (DOF) free to set a possible (not any) orientation for the wrist.

$$P^o_{wrist} = F\begin{pmatrix} L_{Pe} \\ L_{De} \\ L_{Sup} \\ L_{Bi} \end{pmatrix} \tag{3}$$

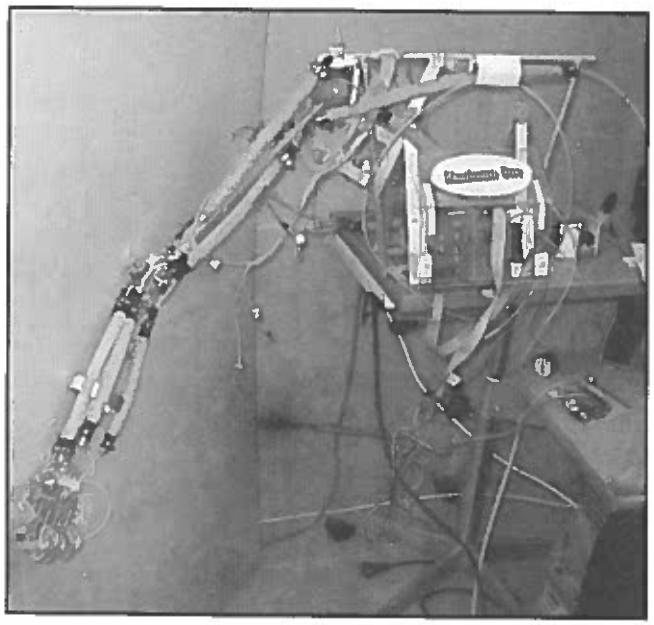

Figure 3: The Anthropomorphic Arm Maximum One, Artificial Intelligent and Robotics Laboratory, Politecnico di Milano, Italy

Now the vectorial field is defined as $F:(R^4 \rightarrow R^3)$, and we can guarantee that for any $L \in U$, where U is a specific subspace of R^4, the field is well defined, and therefore an unique position in the robot workspace is reached.

Lets now consider the direct kinematic problem in a general way considering also the wrist orientation. In order to achieve this we need to do three steps:

- Find the homogeneous matrix that represents orientation and position of the wrist as function of the joint angular positions
- Find the relationship between the angular positions of the joints and the actuators lengths
- Combine the two solutions in order to have orientation and position as a function of the actuators lengths

To solve the first step we can apply a standard method that allows us to find at the end a 4×4 homogeneous coordinates matrix that specifies orientation and position of the wrist; this matrix contains as variables the four joint angular positions $\alpha, \beta, \gamma, \theta$.

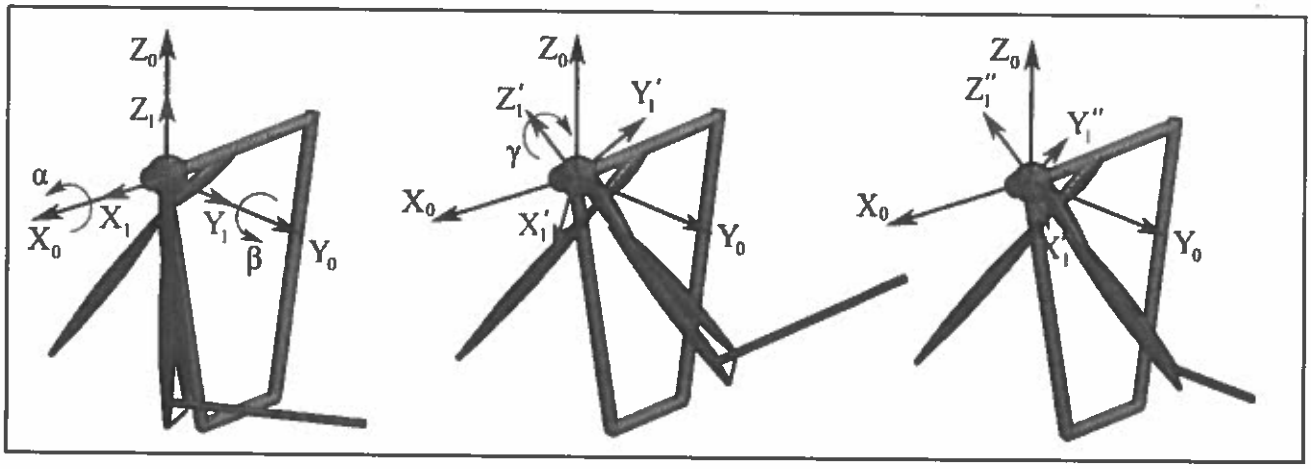

Figure 4: The Reference Systems

As before asserted, thanks to the special actuation architecture for the shoulder we chose, we are able to separate the third rotation along the upper arm link from the first two rotations.

We can represent the first two rotations relatively to the absolute axes (pre- multiplication) X0 and Y0 using a single matrix in homogeneous coordinates $H^0_{0,1'}$.

$$H^0_{0,1'} = \begin{pmatrix} C_\beta & 0 & C_\beta & 0 \\ 0 & 1 & 0 & 0 \\ -S_\beta & 0 & C_\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C_\alpha & -S_\alpha & 0 \\ 0 & S_\alpha & C_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

Then we can perform a rotation along the relative axes $Z_{1'}$ represented by equation 5.

$$H^{1'}_{1',1''} = \begin{pmatrix} C_\gamma & -S_\gamma & 0 & 0 \\ S_\gamma & C_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

Now using the rule of post-multiplication we can represent the reference system connected with the shoulder joint $H^0_{0,1''}$ that depends now on all the three degrees of freedom:

$$H^0_{0,1''} = H^0_{0,1'} \cdot H^{1'}_{1',1''} \tag{6}$$

Moving down along the kinematic chain we encounter the elbow joint, where we locate another reference system with the origin coincident with the elbow center of rotation, the X axis with the same direction of the elbow's rotation axis, and oriented as in figure; this system is connected with the forearm.

We can represent this reference system by the matrix in equation 7

$$H^{1''}_{1'',2'} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C\theta & -S\theta & 0 \\ 0 & S\theta & C\theta & -L_{UpArm} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

The last passage is to perform another translation to obtain the last reference system located in the wrist; we can do this by the operator in equation

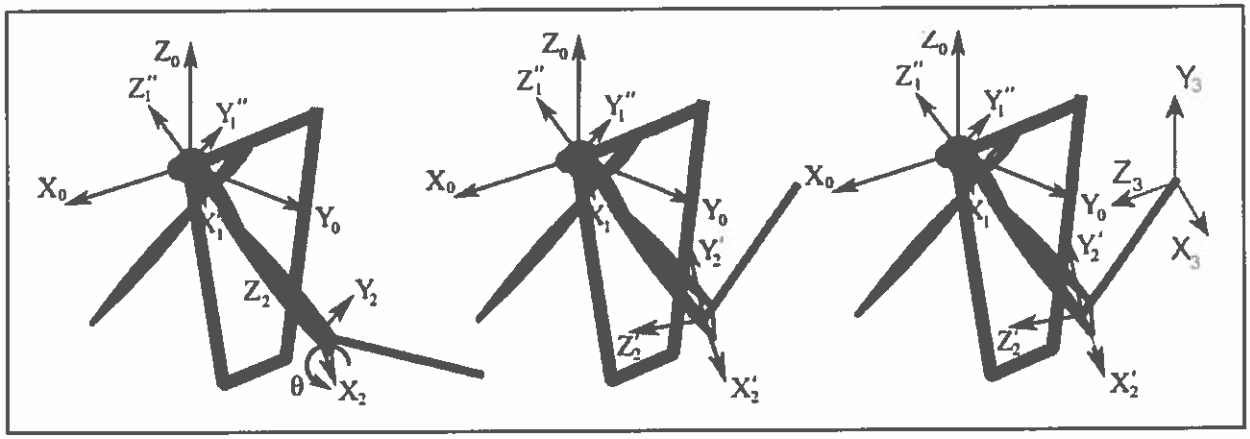

Figure 5: The Elbow Coordinate System

$$H_{2',3}^{2'} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -L_{ForArm} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{8}$$

We can now define the final matrix (equation 9) that represents the wrist reference system (RS3) relative to the absolute reference system (RS0).

$$H_{0,3}^{0} = H_{0,1'}^{0} \cdot H_{1',2'}^{1'} \cdot H_{2',3}^{2'} \tag{9}$$

This matrix is a function of the four angles, $H_{0,3}^{0} = H_{0,3}^{0}(\alpha,\beta,\gamma,\upsilon)$

The second step, in order to solve the direct kinematic problem, requires to find the function (equation 10) that relates the joints angular positions with the actuator lengths.

$$(\alpha,\beta,\gamma,\upsilon) = F(L) \tag{10}$$

Here, to be concise, we explain the step only for the shoulder kinematic, and in the specific case only for the relationship between the angular positions α, β and the three actuators lengths: Ldo, Lpe and Lde. The relationship between γ and the Supraspinatus and Subscapularis actuator lengths and υ and the Biceps and Triceps actuator lengths are more trivial. It also worth mentioning that, thanks to the particular actuation architecture we chose, the angular positions of the joints depend only on certain actuators lengths and in general this relationship is not linear. Equation 11 can be a possible choice for this dependency.

$$\begin{cases} (\alpha,\beta) = F_1(Lpe, Lde) \\ \gamma = F_2(L\sup) \\ \upsilon = F_3(Lbi) \end{cases} \tag{11}$$

Considering three actuators, and assuming that only for two of them we can impose the length, we have three different combinations.

Each actuator has an origin point $O_{actuator}$ (start point) and a insertion point $A_{actuator}$ (end point); assuming that the actuator is not bending this two points define unambiguously also the actuator length $L_{actuator} = \| A_{actuator} - O_{actuator} \|$.

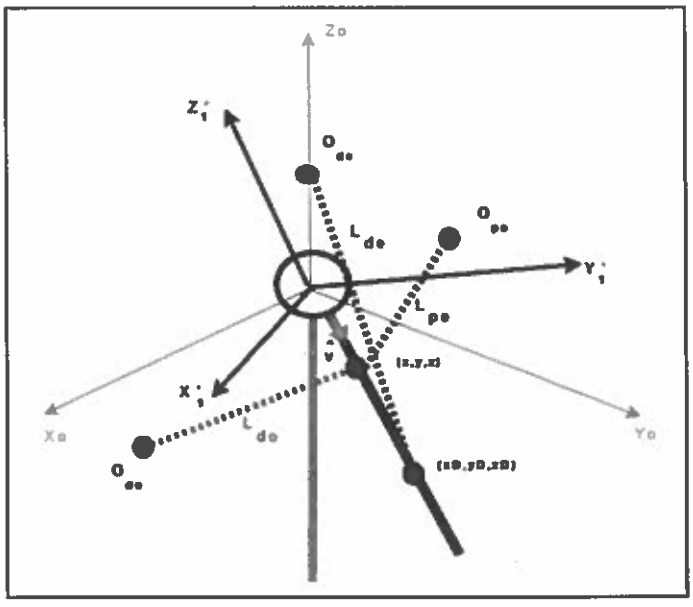

Figure 6: Relationship Between the Deltoid and Pectoral Length and the Angular Rotation α, β along axes X_1 and Y_1

To find the relationship between the angular positions of the joints and the actuators lengths, at first we should find a relationship between a point of the upper arm link, let say the insertion point of the pectoral and dorsal actuators (x, y, z), and the actuator lengths. When this point is defined also the direction of the upper arm link is defined and it will be possible, with simple geometrical considerations, to calculate the angular variables as a function of the actuators lengths.

All the points and parameters we considered in this system of equations (equation 12) are reported in Table 1.

If we impose the length for the pectoralis and deltoid actuators we can write the system in equation 12.

Table 1
Actuators Origin, Insertion and Lengths

Actuator	*Origin Point*	*Insertion Point*	*Actuator Length*
Dorsal	Odo	(x,y,z)	Ldo
Deltoid	Ode	(xD,yD,zD)	Lde
Pectoralis	Ope	(x,y,z)	Lpc

$$\begin{cases} (x-Opc_x)^2+(y-Opc_y)^2+(z-Opc_z)^2-Lpc^2=0 \\ (x-Ode_x)^2+(y-Ode_y)^2+(z-Ode_z)^2-Ldo^2=0 \\ (x-Os_x)^2+(y-Os_y)^2+(z-Os_z)^2-(\|Os-Opc\|)^2=0 \end{cases} \tag{12}$$

The first two equations impose the two actuators lengths as *Lpc* and *Ldo*, and the third equation imposes that after the movement the upper arm link is not changing its length. We can also think at this system in a visual way, where each equation represents a sphere and the solution of the system are the two points that come up from the intersection of the three geometrical entities. In order to find out the solution of this system and get it in a compact and understandable way, we make some substitutions. For example thanks to the fact that the origin of the reference system is located in the center of the shoulder, the vector $O_s = (0, 0, 0)$, also the origin points of each of the actuators under consideration in this system are constant and can be substituted with the real value. The system in equation 12 can be therefore reduced, after some simple passages, to the specific case in equation 13.

$$\begin{cases} (x+20)^2+(y)^2+(z)^2-Lpc^2=0 \\ (x.3)^2+(y.3)^2+(z.3-3)^2-Lde^2=0 \\ (x)^2+(y)^2+(z)^2-(5)^2=0 \end{cases} \tag{13}$$

It is possible now to find the solutions of this system; these are reported in equation 14, and we may note that they have two distinct solutions.

$$\begin{cases} x=\dfrac{Lpc^2-425}{40} \\ y=\pm\dfrac{\sqrt{25.(2754.Lpc^2-16.Lde^4+9cdot(832.Lde^2-147969))-81.Lpe^4}}{360} \\ z=\dfrac{234-Lde^2}{18} \end{cases} \tag{14}$$

Now we can represent a point $p = (x, y, z)$ as a function of the actuators lengths, and finally we can close the direct kinematic model finding the connection between the point P and the angular variables. Looking at picture 6 we can see that this relationship can be found imposing that the versor that defines the direction of the upper arm is equal to the versor $\hat{\upsilon}$ that defines the axes-Z. This last one can be found easily taking the third column of the matrix resulting from the product of the two matrices in equation 4.

$$\begin{pmatrix} \frac{x}{\|p\|} \\ \frac{y}{\|p\|} \\ \frac{z}{\|p\|} \end{pmatrix} = \begin{pmatrix} -s\beta c\alpha \\ -c\alpha \\ -c\beta s\alpha \\ 1 \end{pmatrix} \tag{15}$$

4. INVERSE KINEMATICS

The Inverse Kinematics algorithm we implemented and tested allows the calculation of the actuators lengths vectors L when a target position for the wrist in the robot workspace is assigned. In solving the problem we did not consider the wrist orientation. Anyhow the algorithm we propose does not lose its generality and we think it could be easily extended also to solve the requirement on the wrist orientation.

The algorithm for the Inverse Kinematics problem is based on the gradient descent method. We approached the kinematic problem as an optimization problem. The target function we want to minimize in this case is the distance between the current wrist position and the target position. This function depends on the actuators lengths $D(L)$ and therefore finding a minimum for this function means to find a special configuration for the actuation system that brings the wrist "near" to the target position. In order to find the optimal global solution, we needed a way to avoid local minima. To solve this problem we adapted the simulated annealing method [22] to our specific case in order to find out the best compromise between real time computability and precision of the found solution. In our specific case we need that the virtual temperature decrease very rapidly in order to limit the number of steps to find the global minimum. We need also to note that this does not affect the precision of the solution. Indeed usually the target tra jectory is discretised in a finite number of points N and the optimization process is performed for each of this point. In order to have an evaluation for the error in following the target tra jectory, we can consider the average positioning error as in equation 16 where e_i is the error calculated for the specific point P_i.

$$E = \frac{\sum_{i=1}^{N} e_i}{N} \tag{16}$$

The virtual temperature is changing with the role $T_{t+1} = b \cdot T_t$ where b < 1 and constant. We also modified the annealing algorithm in the sense that the temperature is starting to rise again when the two following condition are verified:

- The minimum obtained has a corresponding wrist position whose distance from the target position is greater than a specified value.
- The new point obtained does not improve the solution. And more precisely the distance dt between the wrist position and the target position satisfies the following condition $0{,}999 \cdot d_{t-1} < d_t < 1{,}001 \cdot d_{t-1}$

This peculiarity of our algorithm allows, with a probability dependent from the number of steps, to exit from a local minimum.

Lets now to explain how we formalized the algorithm that implements the inverse kinematis. When a starting point and a target point are fixed within the arm workspace for the wrist, we need to execute the following main steps:

- Computation of the Arm direct kinematics given as input the actuator length vector L
- Computation of the distance between the current wrist position and the target position
- Computation of the contraction gradient for each actuator

In a more precise way we can codify the algorithm in the pseudo-code of Figure 7.

```
Descent method
-Set Current Wrist Position
-Set Target Wrist Position
-Calculate Distance between Current Position and Target Position
while ( Distance > ε_max and step<max_step )
        for each muscle belonging Sub(i)
                - Decrease its length
                - Calculate the effect on the end-effector position (Direct Kinematics)
                - Calculate the distance between Current Position and Target Position
                - Calculate Gradient
                - Calculate Muscle Velocity
                if (Distance decreases)
                        update muscle length
                        end
        end
        - Increase step
end
```

Figure 7: The Algorithm that Implement the Arm Inverse Kinematics

The algorithm approaches a minimum by taking steps that are proportional to the negative of the gradient. The algorithm ends thanks to two different stopping criteria: the distance between the actual position and the target position is smaller than ϵ_{max}, the number of steps s should not overcome a maximum max_{step}. These conditions are required especially if we need to compute the algorithm in real time; imposing these conditions we can therefore impose an upper bound for the computational time. In the pseudo-code the subset *Sub(i)* represents a couple of actuators chosen with a specific logic from the set of all arm actuators.

In the pseudo-code the incremented velocity for each actuator is calculated by the fast gradient following approach (equation 17), where $0 < \alpha < 1$ and is constant and $\eta(t)$ is a parameters that can assume three different values depending if the solution is improving or becoming worse.

$$\upsilon(t+1) = \eta(t)\upsilon(t) - \alpha \nabla f(p(t)) \tag{17}$$

In our specific case:

- $\eta(t)$ — 1.3 if the approximate solution improve more than the 10%
- $\eta(t)$ — 1.1 if the approximate solution improve less than the 10%
- $\eta(t)$ — 0.5 if the approximate solution is worse than the previous one

5. TRAJECTORIES EXECUTION ON A SIMULATOR

An important capability of a robot arm cooperating with humans is the ability to follow a tra jectory in the Cartesian space. In general this requires to transform Cartesian coordinates into joint coordinates. Each movement requires the coordination of the joints and finally of the artificial muscles.

We have experimented the capability to follow simple trajectories through our simulator, where we input a trajectory in the Cartesian space and obtain the values of the muscle lengths. The simplest tra jectories, namely linear trajectories in the Cartesian space, are easily executed for points at 2.5 cm apart.

To experiment more useful tra jectories we studied circular tra jectories in the Cartesian space. In the experiment, we command four times the execution of the tra jectory, moving the circle center in the working space and changing the plane in which the tra jectory is defined.

We start executing the circumference on a plane parallel to the z-y plane, and with a diameter of 12 cm. The first time the origin is set in the coordinates (-5, 20, -30) and the movement is repeated four times. We can see in Figure 8 the muscle movements and the robot posture during the execution of this tra jectory.

We repeat the experiment and translate the origin along the y axis, moving the center in (-5, 40, -30), and then in (-5, 50, -30). We observe in Figure 9 that the dorsal muscle is working while the Sup and the Sub muscles are not very involved in executing this tra jectory.

We will repeat the same tra jectories on the real prototype robot.

6. EXPERIMENTAL RESULTS ON THE ROBOT

The control system of our robot has been implemented in Matlab 7.0 and Simulink. The software runs on a host PC, where the tra jectories are computed; the host PC is interfaced to a target PC where the operating system xPC-Target runs. The target PC runs the real time routines necessary to interface with the hardware; it sends the commands to the power board in order to move the actuators and receives the sensor signals from the arm.

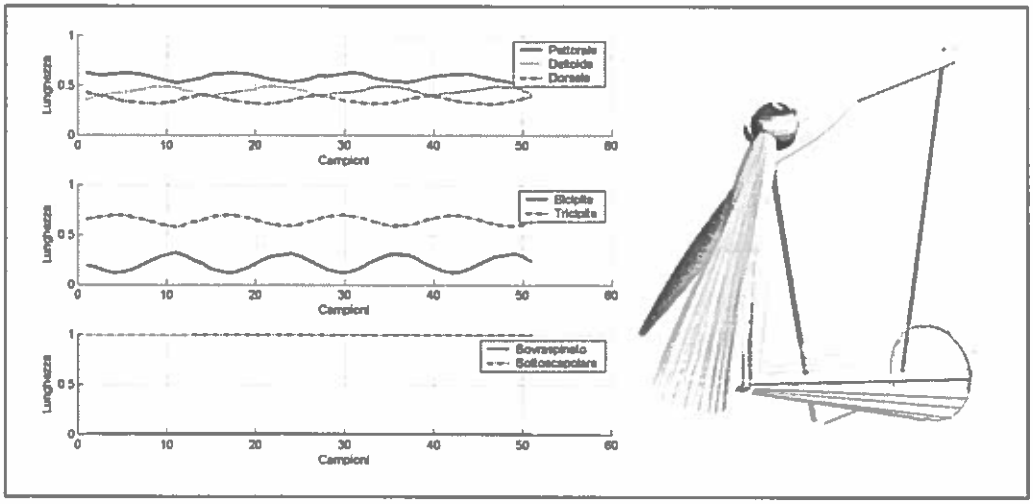

Figure 8: Normalized Actuator Lengths for a Circular Trajectory with Center (-5, 20, -30)

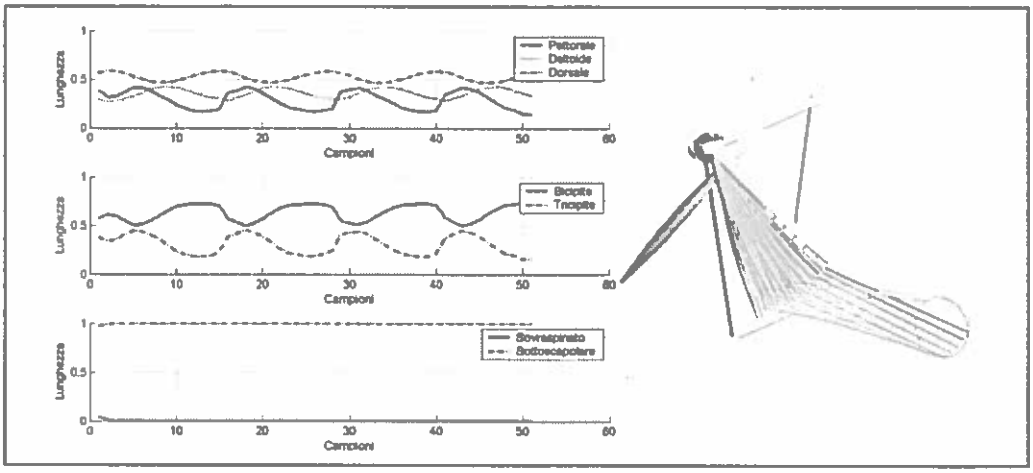

Figure 9: Normalized Actuator Lengths for a Circular Trajectory with Center (-5, 50,-30)

Let us illustrate some quantitative results on the robot. According to the constructive characteristics of the robot arm we defined as a good approximation the value of 1 cm. In the test executed in simulation to check the tra jectory following, we observe that an error of 1 cm is compatible with the execution of the gradient algorithm of 7 steps at least; in this case it calls the direct kinematics 4 times, for the number of independent muscles, at every step. We may observe in fact that when the biceps is active, it acts on a unique degree of freedom, so the triceps can be considered as passive. The same consideration can be extended to other muscles

In Table 2 we compared the computation times, to solve the direct kinematics, of three different methodologies; calculations where done on a 6227 Mips computer (benchmark Dhrystone).

Table 2
Coputational Time

Method Error	*Computational Time (s)*	*Relative Time*	
Fsolve PCG (Matlab)	1,00E-8	3,2	110034
Fsolve LM (Matlab)	1,00E-2	2,5	8620
Direct Kinematic (Closed Form)	1,00E-8	0,00029	1

In the first case, in applying our solutions, we have used the Matlab function "fsolve", that finds the zeros of a function thorough optimization, to partially solve the direct kinematics for the chain that involves the subscapularis and biceps actuators. Fsolve can use a parameter to set the kind of optimization to apply between the (PCG) preconditioned conjugated gradient (first row) and the (LM) Levenberg-Marquardt algorithm [23] (second row). Finally the last row in the table illustrates the solution of the direct kinematics through the joint angles. The solution based on the direct kinematic in closed form, as it is possible to see, is much more faster; nevertheless it does not guarantee that the found solution lies in the admissible space of the muscles, as already discussed.

We analyzed the performances of the algorithm of the gradient descent. In Table 3 we compare the position error of the classic gradient algorithm and of our method, that instead employs simulated annealing and an autoregressive method. The autoregressive method computes the new velocity of each joint during the descent algorithm. We observe that the error in reaching a position is reduced using a faster gradient following.

Table 3
Absolute and Relative Error (Classical Method Relative to Our Method) for the Inverse Kinematic Algorithm Using Classical the Gradient Method and Our Method that Combine Simulated Annealing with Fast Gradient Following

	Gradient Modified	*Classical Gradient*	*Position (x,y,z)*
Error P1 (cm)	1,00E-05	0,0864	(-10,20,15)
Relative Error	1	8640	
Error P2 (cm)	3,30E-05	0,0246	(10,25,-25)
Relative Error	1	745	

We have finally studied the computational complexity of the gradient descent method for inverse kinematics. It requires $K*n \times (n + 1)$ floating points operations for each step, where n is the number of the links and K is the number of products to compute the direct kinematics. If we a-priori limit the number of iterations to 20, we draw in Figure 10 the computational time with different numbers of links.

Since our final task is to execute tra jectories and not reaching a point, we can expect that the next point to solve with the inverse kinematics is quite near to the present point. In this case we are able to find the solution with a

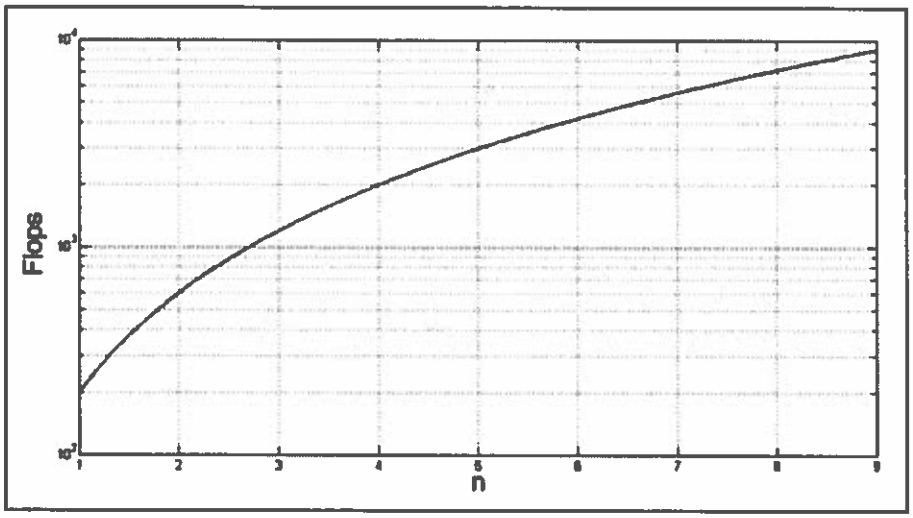

Figure 10: How the Complexity Changes Increasing the Number of Links

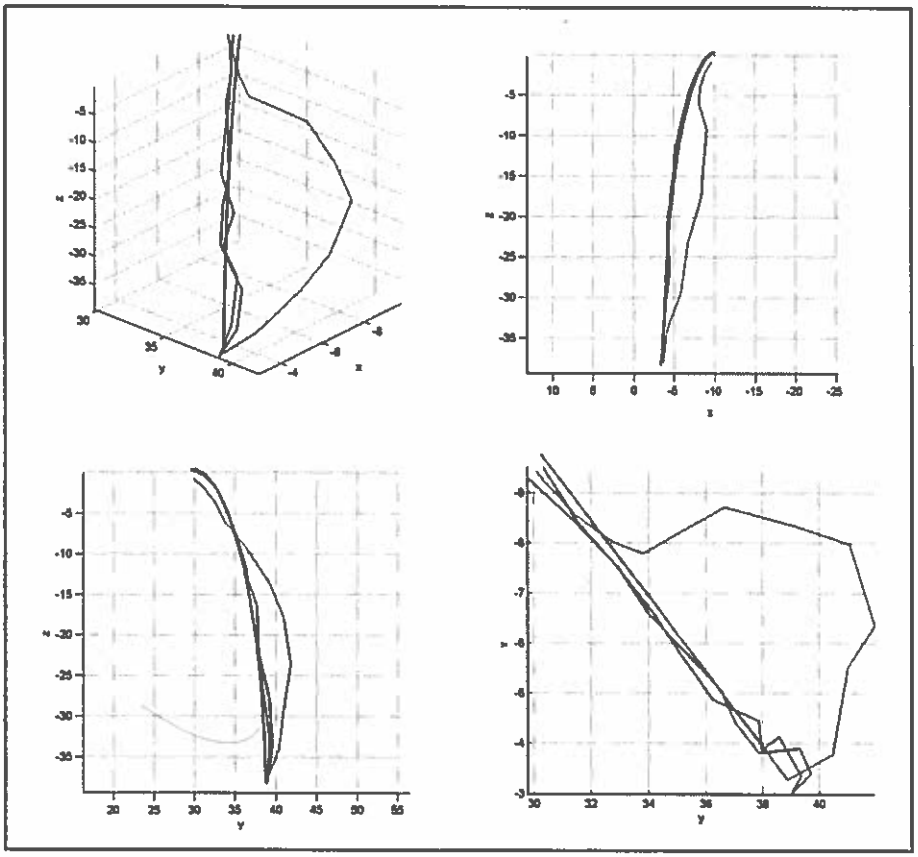

Figure 11: The Trajectories of the Arm in Cartesian Space Using Different Steps (mm)

limited number of steps, so to work in real time. Experimentally we have found that a good number of iterations for the inverse kinematics can be reduced to 11 without loosing the wanted precision of 1 cm.

The tra jectories of the arm in Cartesian space using different steps are illustrated in Figure 11, in green the tra jectory obtained with 3 steps, in blue with 7 steps, in red with 11 steps. As we can see the errors are high if we limit too much the number of steps.

We experimented on the real robot the circular tra jectory already simulated on the robot simulator. The tra jectory is computed in real time from the inverse kinematic module, then data are sent to the target-PC where the real time kernel of Matlab runs the Simulink controller schema.

Figure 12 shows from left to right and from top to bottom the arm execut- ing the tra jectory. The tra jectory is easily executed in 10 s; we reduced the execution time, and noticed some small delay in the output from the inverse kinematics with respect to the controller needs when time is set to 3s, which so can be experimentally considered the minimum time. Finally in Figure 13, we present the plotting of 200 points, in this case the tra jectory is executed many times and it is possible to have an evaluation of the robot repeatability. On the x-axis the variance is 4, 7mm on the y-axis 5, 61mm and on z-axis 4, 49mm.

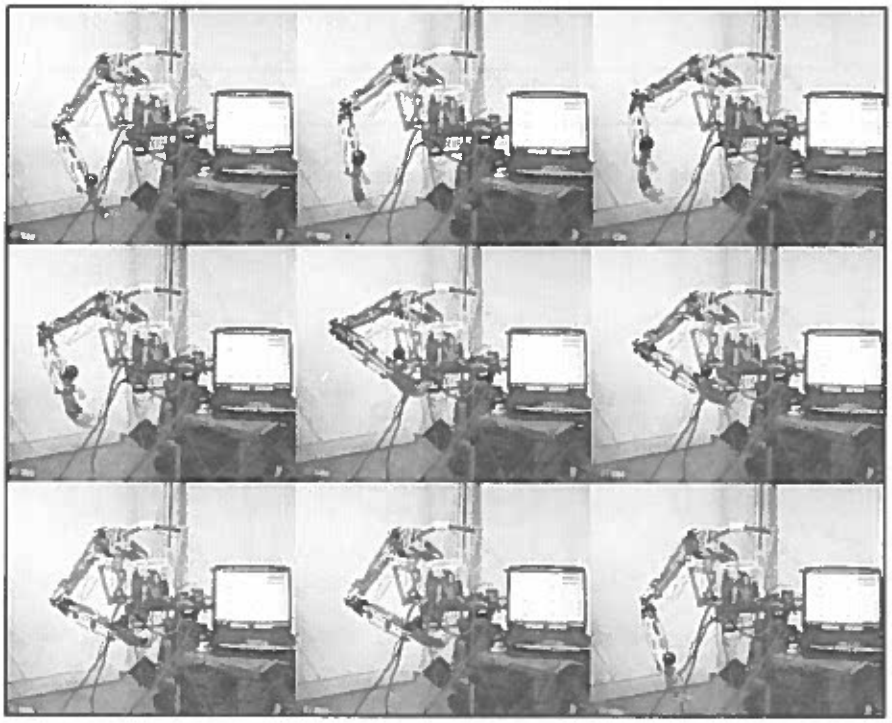

Figure 12: Execution of a Circular Trajectory on the Real Robot

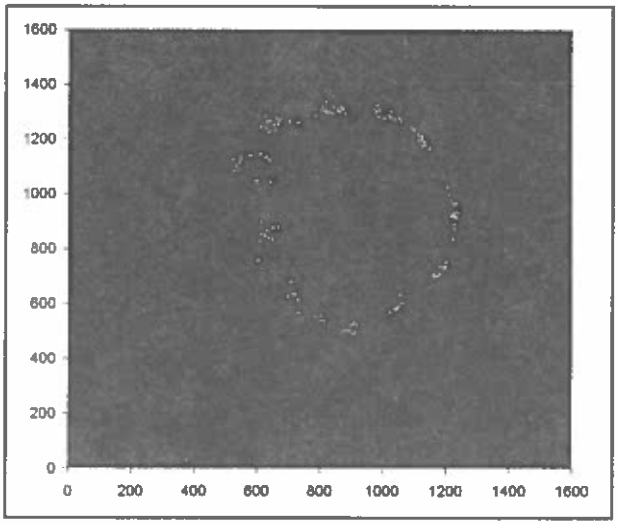

Figure 13: Plotting of 200 Point Recorded During the Circular tra jectory Execution

7. CONCLUSION AND FUTURE WORK

In this paper we described the architecture and the main features of an anthropomorphic artificial arm intended for applications in the field of the Humanoid Robotics. We introduced an innovative actuation system, that emulates the arrangement of the human musculature.

We also developed a novel and general method to approach the direct and inverse kinematics for a humanoid arm. The experiments on the simulator demonstrated the adequacy and efficiency of the proposed algorithms. We were able to solve the inverse kinematics in real time, and to control the seven actuators of the arm to follow given tra jectories.

In particular the inverse kinematics problem was solved using an algorithm based on a fast gradient following method that integrate a simulated annealing technique. This allow us to improve the solutions accuracy and to impose constrains on the computational time.

Furthermore on the real arm we have experimented the solution and found a confirmation of the results, at least if the velocity is quite low.

Those results could be improved in the future through a detailed analysis of the working space. In this case we might be able to use the solution in the joint space, which is faster, and to convert it into the muscles without loosing the real solutions.

A possible future improvement to avoid the calculation of the direct kinematics is to install on the robot a vision system that can perceive the distance between the wrist and the target position. This strategy is very similar to that one performed by humans when it is required to perform a motor path in order to reach a target object. At least during the learning phase, human beings take advantage of the visual feedback to estimate the distance between the object and the hand.

REFERENCES

[1] J. Denavit and R. Hartenberg, "A Kinematic Notation for Lower-pair Mechanisms Based on Matrices," *ASME Journal of Applied Mechanics*, 23, 215–221, 1955.

[2] J. Craig, Introduction to Robotics. Wiley, 1986.

[3] H. C. Axel Schneider and J. Schmitz, "Decentralized control of elastic limbs in closed kinematic chains," *The International Journal of Robotics Research*, 25, 913–930, 2006.

[4] M. Karouia and J. M. Herv´e, "Non-overconstrained 3-dof Spherical Parallel Manipulators of Type: 3-rcc, 3-ccr, 3-crc," *Robotica*, 24, 85–94, 2006.

[5] A. G. Deepak Tolani and N. I. Badler, "Animating Reactive Motion Using Momentum Based Inverse Kinematics," *Computer Animation And Virtual Worlds*, 16, 213–223, 2005.

[6] D. Tolani, A. Goswami, and N. I. Badler, "Real Time Inverse Kinematics Techniques for Anthropomorphic Limbs," *Graph. Models Image Process.*, 62 (5), 353–388, 2000.

[7] E. M. Rosales, J. Q. Gan, H. Hu, and E. Oyama, "A Hybrid Appproach to Inverse Kinematics Modeling and Control of Pioneer 2 Robotic Arms," University of Essex Department of Computer Science, Colchester CO4 3SQ, UK, *Tech. Rep. CSM-412*.

[8] G. Tevatia and S. Schaal, "Inverse Kinematics for Humanoid Robots," *In International Conference on Robotics and Automation (ICRA2000)*, 2000, 294–299. [Online Available: http://www-clmc.usc.edu/publications/T/ tevatia-ICRA2000.pdf]

[9] G. Gini, M. Folgheraiter, "Human-like Reflex Control for An Artificial Hand," *BioSystem Journal*, 76, 1-3, 65–74, August 2004.

[10] K. Kawamura, R. P. II, D. Wilkes, W. Alford, and T. Rogers, "Isac: Foundations in Human-humanoid Interaction." *IEEE Intelligent Systems*, 15 (4), 38–45, July/August 2000.

[11] R. Brooks, C. Breazeal, M. Marjanovic, B. Scassellati, and M. Williamson, "The Cog Project: Building a Humanoid Robot," Computation for Metaphors, Analogy and Agents, Vol. LNCS 1562, 52–87, January 1999.

[12] R. Ambrose, S. Askew, W. Bluethmann, M. Diftler, and M. Lockheed, "Humanoids designed to do Work," *Proc. IEEE-RAS Humanoids 2001 Conference*,Tokyo, Japan, 173–180, November 2001.

[13] C. Atkeson, J. Hale, M. Kawato, S. Kotosaka, F. Pollick, M. Riley, S. Schaal, S. Shibata, G. Tevatia, and A. Ude, "Using Humanoid Robots to Study Human Behaviour," *IEEE Intelligent Systems*, **15**(4), 46–56, 2000.

[14] R. Beer, H. Chiel, R. Quinn, and R. Ritzmann, "Biorobotic Approaches to the Study of Motor Systems," *Current Opinion in Neurobiology*, **8**(6), 777–782, 1998.

[15] P. Dario, E. Guglielmelli, V. Genovese, and M. Toro, "Robot Assistant: Application and Evolution," *Robotic and Autonomous Systems*, **18**(1-2), 225–234, July 1996.

[16] F. Daerden and D. Lefeber, "The Concept and Design of Pneumatic Artificial Muscles," *International Journal of Fluid Power*, **2**(41–50), 2001.

[17] G. K. Klute and B. Hannaford, "Accounting for Elastic Energy Storage in Mckibben Artificial Muscle Actuators," *ASME Journal of Dynamic Systems, Measurement, and Control*, **122** (2), 386–388, 2000.

[18] C. P. Chou and B. Hannaford, "Measurement and Modeling of Mckibben Pneumatic Artificial Muscles," *IEEE Transactions on Robotics and Automation*, **12**(1), 90–102, 1996.

[19] G. Tonietti and A. Bicchi, "Adaptive Simultaneous Position and Stiffness Control for a Soft Robot Arm," *Proc. IEEE Int. Symp. Intelligent Robots and Systems*, 1992–1997, October.

[20] B. Tondu and P. Lopez, "Modeling and Control of Mckibben Artificial Muscle Robot Actuators," *IEEE Control Systems Magazine*, **20**(2), 15–38, April 2000.

[21] S. Ashley, "Artificial Muscles," Scientific American, October 2003.

[22] C. G. S. Kirkpatrick and M. Vecchi, "Optimization by Simulated Annealing," *Science*, **220**(4598), 671–680, 1983.

[23] J. J. Mor´e, "The Levenberg-marquardt Algorithm: Implementation and Theory," *Numerical Analysis*, ed. G. A. Watson, Lecture Notes in Mathematics, **630**, 105–116, 1977.