2009 International Conference on Electrical, Communications, and Computers

Forward and Inverse Kinematics for a Small-Sized Humanoid Robot

J.M. Ibarra Zannatha, R. Cisneros Limón
Departamento de Control Automático
Centro de Investigación y Estudios Avanzados
del Instituto Politécnico Nacional
E-mail: {jibarra, rcisneros}@ctrl.cinvestav.mx

***Abstract*—This paper examines the forward and inverse kinematics of a small-sized low-cost commercial humanoid robot providing closed form solutions for both kinematic problems. The analyzed robot has 16 degrees-of-freedom: five for each leg and three for each arm, actuated by sixteen digital servomotors, like those used in aeromodelism. Each leg is a serial kinematic chain that uses three degrees of freedom to control the position of the foot, and two more to orient it, resembling a human ankle. Unfortunately, this ankle has been designed with an offset that prevents a direct decoupling of the position and orientation of the foot, situation that generally leads to the lack of a closed form solution in most cases. The proposed solution deals with this problem; it uses a geometrical approach that effectively solves the inverse kinematic problem by means of an analytical algorithm.**

I. Introduction

A rigid multibody system consists of a set of rigid objects, called links, joined together by joints. Simple types of joints include revolute and prismatic joints, also called rotational and translational joints respectively. Well-known applications of rigid multibodies include robots (robotic arms, legged robots and humanoids), and virtual skeletons of human and animals for animation in computer graphics; all of them are complex mechanisms composed by serial kinematic chains with some degree of redundancy.

The most interesting movement tasks for this kind of redundant multibody systems, such as walking, are defined in the task space. This space is different from the joint space where motor commands must be issued. Hence, movement planning requires appropriate coordinate transformations from task to joint space before motor commands can be calculated [1] [2] [3].

We will focus on the case where movement plans are given as task space kinematic trajectories on humanoids with many redundant degrees-of-freedom (DOFs) (Figure 1). The transformation from kinematic plans in task space coordinates to joint coordinates is the classic Inverse Kinematics (IK) problem, a problem that arises from the fact that inverse transformation is often ill-posed. Therefore, to control the movement of a rigid multibody, it is common to use IK, where it is presumed that specified points on the links, called *end effectors*, are assigned *target positions*. To solve the IK problem, we must find settings for the joint angles, so that the resulting configuration of the multibody places each end effector at its target position. More general formulations of IK allow orientation or directional goals [1] [2].

If we define the intrinsic coordinates of a manipulator as the vector of joint angles $\boldsymbol{\theta} \in \mathbb{R}^n$, and the position and orientation of the manipulator's end effector as the vector $\boldsymbol{x} \in \mathbb{R}^m$, the forward kinematic function is generally written as:

$$\boldsymbol{x} = f(\boldsymbol{\theta}) \tag{1}$$

However, we need the inverse relationship:

$$\boldsymbol{\theta} = f^{-1}(\boldsymbol{x}) \tag{2}$$

For redundant manipulators, $n > m$, solutions to Equation (2) are usually non-unique (excluding the degenerate case where no solutions exist at all), and even for $n = m$ multiple solutions can exist [4]. Therefore, IK algorithms need to address how to determine a particular solution to (2) in the case of multiple solutions.

The most simple case to find a closed-form solution for the IK problem, is when the kinematic chain has 6 DOF, and the position is responsibility of the first three DOF, while the orientation is given by the three remaining DOF. Even for singularity-free mechanisms, the presence of some kinematic parameters (offsets) prevents decoupling of the position and orientation in the manipulator inverse kinematics problem. Therefore, closed-form solutions are difficult, if not impossible, to find.

There are several methods for solving IK problems, that derive originally from robotics applications. We have methods based on geometric relationships between task and joint coordinates, and methods based on kinematic decoupling. We can use also the Paden-Kahan method based on Screw Theory applied to non-redundant robots with 6 DOF [5]. But when the mechanism has singularities, offsets or redundancies, heuristic methods like freezing DOF to eliminate redundancies have been proposed. However, redundant DOF are useful to optimize additional constraints

978-0-7695-3587-6/09 $25.00 © 2009 IEEE
DOI 10.1109/CONIELECOMP.2009.50

like manipulability, force constraints, obstacle avoidance, etc. Thus, it is interesting to solve the inverse problem Equation (2) by imposing an optimization criteria:

$$g = g(\boldsymbol{\theta}) \tag{3}$$

Where g is usually a convex function that has a unique global minimum.

There are two generic approaches to solving IK problems with optimization criteria [1]. Global methods find an optimal path of $\boldsymbol{\theta}$ with respect to the entire trajectory, usually in computationally expensive off-line calculations. In contrast, local methods, which are feasible in real time, only compute an optimal change in $\boldsymbol{\theta}$ ($\Delta\boldsymbol{\theta}$) for a small change in $\boldsymbol{x}$ ($\Delta\boldsymbol{x}$) and then integrate $\Delta\boldsymbol{x}$ to generate the entire joint space path. Resolved Motion Rate Control (RMRC) is one of such local methods that uses the Jacobian J of the forward kinematics to describe a change of the end effector's position as:

$$\boldsymbol{x'} = J(\boldsymbol{\theta})\boldsymbol{\theta'} \tag{4}$$

If J is square, $m = n$, and non-singular, this equation can be solved for $\boldsymbol{\theta'}$ by taking the inverse of J. Redundant manipulators, $n > m$, require the use of additional constraints to obtain a unique inverse. In this case we can use $J^{\dagger}$, the Moore-Penrose pseudo-inverse

$$\boldsymbol{\theta'} = J^{\dagger}\boldsymbol{x'} = J^{T}(JJ^{T})\boldsymbol{x'} \tag{5}$$

Which minimizes the norm of $\boldsymbol{\theta'}$. A more general form of optimization with pseudo-inverses by minimizing an explicit objective function g in the null space of J has been suggested:

$$\boldsymbol{\theta'} = J^{\dagger}\boldsymbol{x'} - \alpha(I - J^{\dagger}J)\partial g/\partial\boldsymbol{\theta} \tag{6}$$

As a general problem, pseudo-inverse methods are not conservative. For example, a closed path in task space does not guarantee a closed path in joint space. Additionally, it is not always easy to determine the constant α in (6) which controls the influence of the optimization function g. These problems motivated the Extended Jacobian Method (EJM), which is an optimization method that obtains a conservative RMRC solution. The goal of the EJM is to zero the gradient of $g(\boldsymbol{\theta})$ in the null space of the Jacobian and track this optimal solution. For this purpose, the gradient of the optimization criterion, $\partial g/\partial\boldsymbol{\theta}$, is projected onto the null space basis vectors of J, given by the columns of the singular value decomposition of J [1].

Other methods to solve the IK problem are cyclic coordinate descent methods, pseudo-inverse methods, Jacobian transpose methods, the Levenberg-Marquardt damped least squares methods, quasi-Newton and conjugate gradient methods, methods based on Linear Programming [6], and many methods based on Soft Computing techniques such as Genetic Algorithms, Fuzzy Systems, Neural Nets and Artificial Intelligence [7] [8]. Finally, there are some special tools used for controlling the motion of a rigid multibody system, like IKAN software (Inverse Kinematics using Analytical Methods) developed at the University of Pennsylvania [3].

This paper focuses on applications of IK for the ROBONOVA humanoid like simulating the task of walking, and specifying the position and orientation of its feet in task space coordinates.

TABLE I
RANGE OF MOTION OF EACH ARTICULATION

Joint		Range of Motion
Hip (Frontal)	Left	$-95^{\circ} \leq \theta_{dl1} \leq 10^{\circ}$
	Right	$-10^{\circ} \leq \theta_{dr1} \leq 95^{\circ}$
Hip (Sagital)	Left	$-72.1^{\circ} \leq \theta_{dl2} \leq 87.9^{\circ}$
	Right	$-87.9^{\circ} \leq \theta_{dr2} \leq 72.1^{\circ}$
Knee (Sagital)	Left	$-13.6^{\circ} \leq \theta_{dl3} \leq 161.4^{\circ}$
	Right	$-161.4^{\circ} \leq \theta_{dr3} \leq 13.6^{\circ}$
Ankle (Sagital)	Left	$-105^{\circ} \leq \theta_{dl4} \leq 50^{\circ}$
	Right	$-50^{\circ} \leq \theta_{dr4} \leq 105^{\circ}$
Ankle (Frontal)	Left	$-45^{\circ} \leq \theta_{dl5} \leq 10^{\circ}$
	Right	$-10^{\circ} \leq \theta_{dr5} \leq 45^{\circ}$

II. ROBONOVA-I HUMANOID ROBOT

ROBONOVA is a small-sized commercial humanoid robot developed by HITEC. It is distributed at an affordable price. However, it has some limitations regarding its mechanical design, actuators, computing power and sensorial capacity.

This humanoid, shown in Figure 1, is 31 cm tall and weights 1.3 kg. Its structure consists of sixteen digital servo motors, joined by anodized aluminum servo brackets and nylon servo cases that provide a strong and durable exoskeleton. It has 16 DOF: five for each leg and three for each arm. The first two DOF of each leg correspond to the two human's hip movements in the frontal and sagital planes. The third joint resembles the human's knee movement, and the last two DOF correspond to the two human ankle movements in the frontal and sagital planes.

Each servo motor has an associated range of motion, so that each joint has a lower and upper limit. This situation is summarized on Table I.

Fig. 1. Robonova-I Humanoid Robot

III. Humanoid Forward Kinematics

The first step to obtain the Forward Kinematics (FK) of ROBONOVA is to construct the kinematic diagram shown in Figure 2. In this diagram, joints are represented by cylinder-shaped icons, in which an arrow represents its positive sense of motion. Small spheres represent some important reference points like the center of gravity (*COG*) of the body (B), the neck (N), both hands and feet (Hs and Fs, where $s = \{L, R\}$ stands for left or right), and auxiliary points placed at shoulder-level (U for *Up*) and at pelvis-level (D for *Down*).

The world reference frame is oriented in such a way that the y axis points up and the x axis points in the movement direction of the robot. The z axis is established to complete a right-hand frame. Initially, the body frame is coincident with the world reference frame, as well as the feet frames. Each joint is labeled according to the following nomenclature: $J_{fs\#}$, where:

- f: u (*up*) for arms or d (*down*) for legs.
- s: l (*left*) or r (*right*).
- $\#$: A number starting at 1 at the joint next to the body frame (base frame) and increasing until reaching the last joint in each kinematic chain.

Links follow a similar nomenclature, $L_{fs\#}$, where the subscript has the same meaning as for the joints. Joints and links with the same subscript are related. Furthermore, each link has an associated frame attached to it, placed according to the Denavit-Hartenberg convention. Each joint's axis is labeled in the same manner as the link (or reference point) to whom it is attached. In order to not transgress both Denavit-Hartenberg's rules, we have placed extra frames at the auxiliar reference points. Recall that these rules are:

DH1 The axis x_i is perpendicular to the axis z_{i-1}.
DH2 The axis x_i intersects the axis z_{i-1}.

Additionally, the kinematic diagram has some extra useful information, like link dimensions denoted by $l_{f\#}$, where side information was omitted for symmetry reasons. Other information included, is the motion range of each joint as it is reported on Table I.

The Denavit-Hartenberg parameters for each leg are shown in Table II, where $s = \{l, r\}$ and $x = \{1, 2\}$ respectively.

Based on these parameters, we constructed the so called A-matrices, which when multiplied together, yield the following homogeneous transformation, representing each foot's pose:

$$T_B^{FX} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

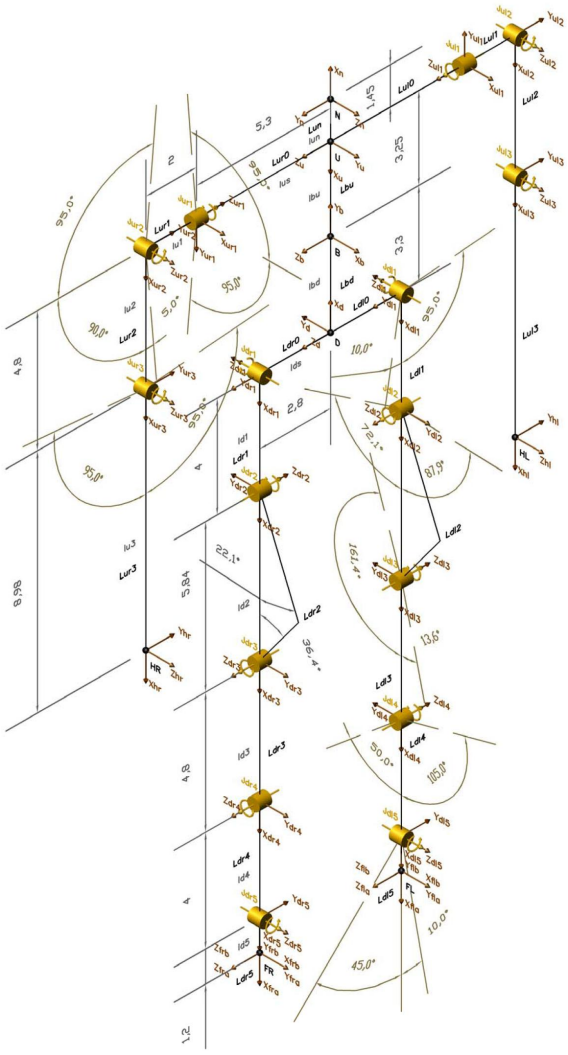

Fig. 2. Robonova's Kinematic Diagram

TABLE II
LINK PARAMETERS FOR EACH LEG

Link	a_i	α_i	d_i	θ_i
L_{bd}	$-l_{bd}$	$0°$	0	$90°$
L_{ds0}	0	$90°$	$-(-1)^{x-1}l_{ds}$	$180°$
L_{ds1}	l_{d1}	$-(-1)^{x-1}90°$	0	θ_{ds1}
L_{ds2}	l_{d2}	$180°$	0	θ_{ds2}
L_{ds3}	l_{d3}	$0°$	0	θ_{ds3}
L_{ds4}	l_{d4}	$(-1)^{x-1}90°$	0	θ_{ds4}
L_{ds5a}	l_{d5}	$90°$	0	θ_{ds5}
L_{ds5b}	0	$0°$	0	$90°$

where,

$$
\begin{aligned}
r_{11} &= c_{2-3-4} \\
r_{21} &= (-1)^{x-1} c_1 s_{2-3-4} \\
r_{31} &= -(-1)^{x-1} s_1 s_{2-3-4} \\
r_{12} &= -(-1)^{x-1} s_{2-3-4} c_5 \\
r_{22} &= c_1 c_{2-3-4} c_5 + s_1 s_5 \\
r_{32} &= -(-1)^{x-1} s_1 c_{2-3-4} c_5 + c_1 S_5 \\
r_{13} &= (-1)^{x-1} s_{2-3-4} s_5 \\
r_{23} &= -c_1 c_{2-3-4} s_5 + s_1 c_5 \\
r_{33} &= s_1 c_{2-3-4} s_5 + c_1 c_5 \\
d_x &= (-1)^{x-1} l_{d5} s_{2-3-4} c_5 + (-1)^{x-1} l_{d4} s_{2-3-4} \\
&\quad + (-1)^{x-1} l_{d3} s_{2-3} + (-1)^{x-1} l_{d2} s_2 \\
d_y &= -l_{d5} \left(c_1 c_{2-3-4} c_5 + s_1 s_5 \right) - l_{d4} c_1 c_{2-3-4} \\
&\quad - l_{d3} c_1 c_{2-3} - l_{d2} c_1 c_2 - l_{d1} c_1 - l_{bd} \\
d_z &= l_{d5} \left(s_1 c_{2-3-4} c_5 - c_1 s_5 \right) + l_{d4} s_1 c_{2-3-4} \\
&\quad + l_{d3} s_1 c_{2-3} + l_{d2} s_1 c_2 + l_{d1} s_1 - (-1)^{x-1} l_{ds}
\end{aligned}
$$

It's worth to notice that when writting the above equation, we used the following shorthand notation: $c_{\#} := \cos(\theta_{ds\#})$ and $s_{\#} := \sin(\theta_{ds\#})$. As well as,

$$\theta_{ds(\#_1 \pm \#_2 \pm \ldots \pm \#_n)} := \theta_{ds\#_1} \pm \theta_{ds\#_2} \pm \ldots \pm \theta_{ds\#_n}$$

Once we get the above homogeneous transformation, we can represent the orientation of the foot by giving three Euler angles, which are defined as follows:

Given that the initial orientation of the foot coincides with that of the world reference frame, we suppose a first rotation of θ_s about the z axis (*pitch*), followed by a second rotation of θ_f about the actual x axis (*roll*), and then a third rotation of θ_t about the actual y axis (*yaw*). Then, the resulting rotation matrix is given by

$$
\begin{aligned}
R &= R_z R_x R_y \\
&= \begin{bmatrix} c_s c_t - s_s s_f s_t & -s_s c_f & c_s s_t + s_s s_f c_t \\ s_s c_t + c_s s_f s_t & c_s c_f & s_s s_t - c_s s_f c_t \\ -c_f s_t & s_f & c_f c_t \end{bmatrix}
\end{aligned} \tag{8}
$$

Once we get the above rotation matrix from the defined Euler angles, we can extract which set of angles yield a specific rotation matrix, as follows:

$$\theta_f = \sin^{-1} R_{3,2} \tag{9}$$

$$\theta_s = \tan^{-1} \frac{-R_{12}}{R_{22}} \tag{10}$$

$$\theta_t = \tan^{-1} \frac{-R_{31}}{R_{33}} \tag{11}$$

IV. Humanoid Inverse Kinematics

FK is a coordinate mapping from joint space to task or Cartesian space. For serial manipulators, the FK given by Equation (1) is straightforward. The inverse mapping expressed by Equation (2), is an ill-posed problem difficult to solve because the system is coupled, nonlinear, and multiple solutions generally exist; redundancy and some mechanical design parameters complicate the problem.

A. The Kinematic Decoupling Problem

Although the general IK problem is quite difficult, it turns out that when the axes of the last three joints intersect at a point (spherical wrist) it is possible to decouple the IK problem into two simpler problems, known as inverse position kinematics (IPK) and inverse orientation kinematics (IOK). In those cases, a closed-form solution exists. This solution can be found by decoupling the Cartesian position and orientation of the end effector, because its position is a function only of the first three joint angles. After solving for position, the relative wrist orientation can be found and the last angles can be calculated.

When wrist (or ankle) has an offset separating its coordinate frames (Figure 3) [9], the situation prevents decoupling of position and orientation, which complicates the solution of the IPK problem. The offset wrist represents a non-zero vector ${}^{\boldsymbol{F}}\boldsymbol{P_H}$ and the Cartesian position is a function of all joint coordinates. Three more equations are obtained from the orientation command, resulting in six non-linear equations coupled with six unknowns. A closed-form solution may not exist.

The following equation is an attempt to decouple both the position and orientation:

$${}^{\boldsymbol{B}}\boldsymbol{P_F} = {}^{\boldsymbol{B}}\boldsymbol{P_H} - {}^{B}_{F}R\,{}^{\boldsymbol{F}}\boldsymbol{P_H} \tag{12}$$

But this one fails, because ${}^{\boldsymbol{F}}\boldsymbol{P_H}$ depends on the last joint coordinates, still unknown [9].

B. Geometrical Approach

The proposed algorithm solves the IK problem for ROBONOVA, whose kinematic diagram is shown in Figure 2, by means of a geometrical approach, based on vectors and planes, regardless of the abscense of kinematic decoupling. Let us pose the IK problem as follows: Given the actual position and orientation of the body, and the desired position and orientation of the considered foot (left or right), we must find the set of five joint coordinates of the corresponding

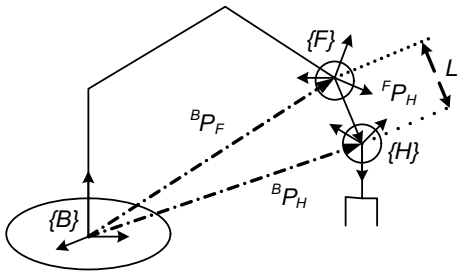

Fig. 3. Manipulator with Offset Wrist

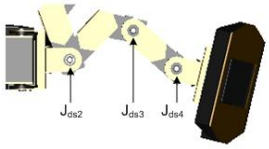

Fig. 4. Placement of leg joints J_{ds2} to J_{ds4}

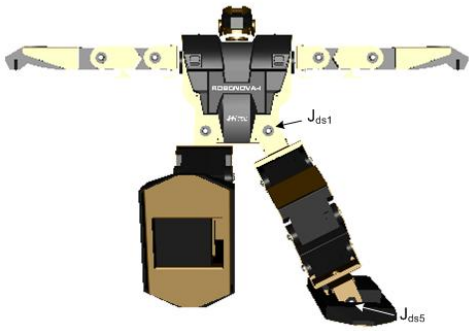

Fig. 5. Placement of leg joints J_{ds1} and J_{ds5}

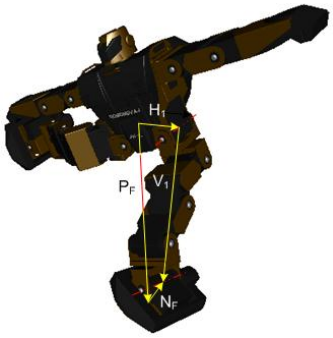

Fig. 6. Initially defined vectors needed by the proposed algorithm

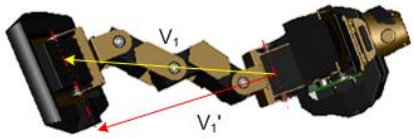

Fig. 7. Projection of V_1 onto the frontal plane

leg that produces such configuration. Once this problem is solved we can use its solution for any walking stage, no matter if the corresponding leg is playing the role of support leg or floating leg.

It is worth to notice two useful facts: i) Axes of joints J_{ds2}, J_{ds3} and J_{ds4} are parallel, so the corresponding links lie on the same plane (Figure 4), and ii) Joints J_{ds1} and J_{ds5} orient such plane (Figure 5).

Without loss of generality, let us suppose that the body is positioned at the origin and its orientation is coincident with the world reference frame. The inverse of the required transformation is then applied to the position and orientation of the foot in order to describe its pose in the new body frame.

Each leg has 5 DOF, three corresponding to the position of the foot, and two corresponding to its orientation, specified by the pitch and roll angles of the foot, while the yaw angle is configuration dependant. This choice is based on the resulting matrix shown in (8). The second column of this matrix represents the y-axis of the frame fixed to the foot. This axis originally points up, so it is coincident with the foot's normal and it does not depend on the yaw angle θ_t, so we can use it to solve the position of the J_{ds5} joint, recalling that this second column is in fact a unit vector, which when multiplied by the L_{ds5}-link's length (l_{d5}), it will represent the vector $\boldsymbol{N_f}$ shown in Figure 6:

$$\boldsymbol{N_f} = l_{d5} \cdot \begin{bmatrix} -s_s c_f & c_s c_f & s_f \end{bmatrix}^T \tag{13}$$

The position of the J_{ds1} joint is initially known since it is in fact a displacement, described in the body frame. We define a vector $\boldsymbol{H_1}$ that points from the origin of the body frame to this joint, according to the information given in Figure 2. This vector is also shown in Figure 6 and it is defined as:

$$\boldsymbol{H_1} = \begin{bmatrix} 0 & -l_{bd} & -(-1)^{x-1} l_{ds} \end{bmatrix}^T \tag{14}$$

The remaining vectors shown in Figure 6 are the foot's position $\boldsymbol{P_f}$ (given as a data) and a vector $\boldsymbol{V_1}$ pointing from J_{ds1} to J_{ds5} calculated as:

$$\boldsymbol{V_1} = \boldsymbol{P_f} + \boldsymbol{N_f} - \boldsymbol{H_1} \tag{15}$$

In order to calculate joint coordinate θ_{ds1} we project $\boldsymbol{V_1}$ onto the frontal plane of the robot (Figure 7).

Once this is done, θ_{ds1} is calculated as the angle between the projected vector $\boldsymbol{V_1'}$ and the sagital plane (Figure 8). Or equivalentely, as the angle between the projected vector $\boldsymbol{V_1'}$ and a vertical vector pointing down ($-\hat{\boldsymbol{j}}$). Recall that the body frame is coincident with the world reference frame. That is,

$$\theta_{ds1} = \cos^{-1} \frac{\boldsymbol{V_1'} \cdot (-\hat{\boldsymbol{j}})}{|\boldsymbol{V_1'}|} \tag{16}$$

It's worth to mention that Equation (16) is incomplete. The argument of $\cos^{-1}$ is always a positive angle, so that (16) will yield the same result no matter which side of the sagital plane the vector $\boldsymbol{V_1'}$ is. In order to include the missing sign, we should modify (16) as follows:

$$\theta_{ds1} = -\text{sgn}(\boldsymbol{V_1'} \cdot \hat{\boldsymbol{k}}) \cos^{-1} \frac{\boldsymbol{V_1'} \cdot (-\hat{\boldsymbol{j}})}{|\boldsymbol{V_1'}|} \tag{17}$$

By doing this, the sign of θ_{ds1} is obtained from the z-component of $\boldsymbol{V_1'}$.

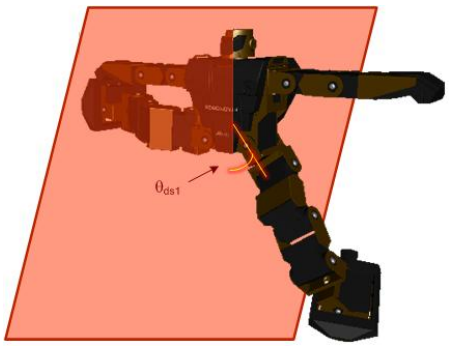

Fig. 8. θ_{ds1} definition with respect to the sagital plane of the robot

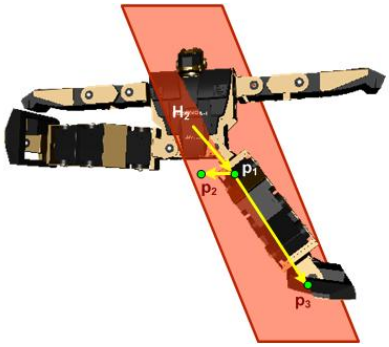

Fig. 9. Leg's characteristic plane

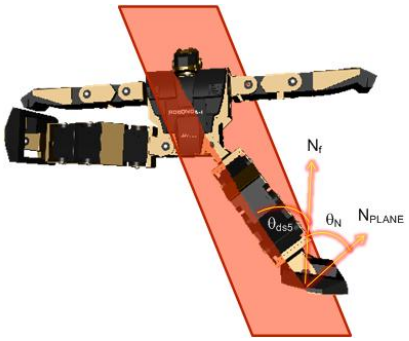

Fig. 10. θ_{ds5} related to the angle between normals

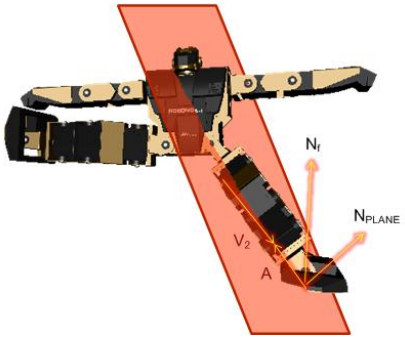

Fig. 11. Definition of vector $\boldsymbol{V_2}$

Once θ_{ds1} is known, it is possible to compute the position of the J_{ds2} joint, and define a vector $\boldsymbol{H_2}$ that points from the origin of the body frame to this joint (Figure 9):

$$\boldsymbol{H_2} = \boldsymbol{H_1} + \begin{bmatrix} 0 & -l_{d1}c_1 & -l_{d1}s_1 \end{bmatrix}^T \tag{18}$$

With three points, it is possible to completely define a plane in space, characterized by its normal vector. Let us define the plane perpendicular to the axes of joints J_{ds2}, J_{ds3} and J_{ds4}:

Knowing J_{ds2} position, we can use it as the first point, $\boldsymbol{p_1}$. The second point, $\boldsymbol{p_2}$, can be calculated as the sum of the previous one with an offset in the movement direction, which preserves its z-component. Calculated in this way, the second point will always lie on the defined plane (Figure 9). Finally, the third point, $\boldsymbol{p_3}$, can be selected as the position of J_{ds5}, since this joint also lies on the defined plane (Figure 9). Indeed, the axes of J_{ds1} and J_{ds5} are parallel to this plane. Having done that, the normalized plane's normal vector N_{plane} is calculated as follows:

$$\boldsymbol{N}_{\text{plane}} = \frac{(\boldsymbol{p_2} - \boldsymbol{p_1}) \times (\boldsymbol{p_3} - \boldsymbol{p_1})}{\|(\boldsymbol{p_2} - \boldsymbol{p_1}) \times (\boldsymbol{p_3} - \boldsymbol{p_1})\|} \tag{19}$$

Notice from Figure 10 that finding θ_{ds5} is the same problem as finding the angle between the foot's normal vector and the plane previously defined. But, in order to calculate this angle we first find the angle between both normals:

$$\theta_N = \cos^{-1} \frac{\boldsymbol{N_f} \cdot \boldsymbol{N}_{\text{plane}}}{l_{d5}} \tag{20}$$

And θ_{ds5} is calculated as θ_N's's complementary angle:

$$\theta_{ds5} = \sin^{-1} \frac{\boldsymbol{N_f} \cdot \boldsymbol{N}_{\text{plane}}}{l_{d5}} \tag{21}$$

Using Equation (21) has the advantage of providing the angle's sign.

Once this is done, we calculate the position of the J_{ds4} joint. This is done by projecting the foot's normal angle onto the leg's characteristic plane (Figure 11).

A vector's projection onto an arbitrary plane can be obtained by first projecting it onto the plane's normal and then substracting the result from the original vector. Therefore, we calculate the foot's normal projection onto the leg's characteristic plane $\boldsymbol{N'_f}$ as follows:

$$\boldsymbol{N'_f} = \boldsymbol{N_f} - (\boldsymbol{N_f} \cdot \boldsymbol{N}_{\text{plane}})\boldsymbol{N}_{\text{plane}} \tag{22}$$

And then, we obtain a vector pointing from the foot's base to the ankle $\boldsymbol{A}$ (Figure 11) calculated as:

$$\boldsymbol{A} = \boldsymbol{N_f} + l_{d4}\frac{\boldsymbol{N'_f}}{\left\|\boldsymbol{N'_f}\right\|} \tag{23}$$

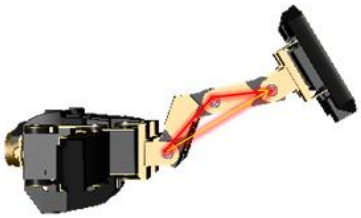

Fig. 12. Reduced problem: a two-link planar leg

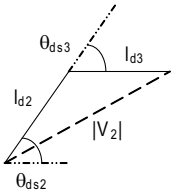

Fig. 13. Two-link planar leg diagram

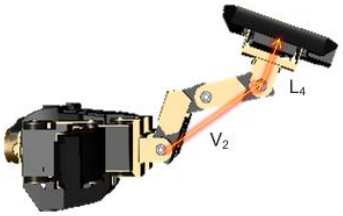

Fig. 14. Definition of the auxiliar vector $\boldsymbol{L_4}$

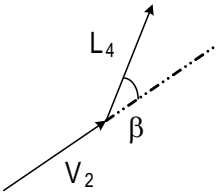

Fig. 15. Definition of β angle

Finally, let us define a vector $\boldsymbol{V_2}$ that points from J_{ds2} to J_{ds4}, also shown in Figure 11, expressed as:

$$\boldsymbol{V_2} = \boldsymbol{P_f} + \boldsymbol{A} - \boldsymbol{H_2} \tag{24}$$

Now, the whole problem is reduced to the classical two-link planar robot (Figure 12) whose kinematic diagram is presented in Figure 13. The solution to this problem is [10]:

$$D = \cos\theta_{ds3} = \frac{\|\boldsymbol{V_2}\|^2 - l_{d2}^2 - l_{d3}^2}{2 l_{d2} l_{d3}} \tag{25}$$

$$\theta_{ds3} = \arctan2\left(-\sqrt{1-D^2}, D^2\right) \tag{26}$$

$$\theta_{ds2} = \arctan2\left(V_{2,x}, \sqrt{V_{2,y}^2 + V_{2,z}^2}\right) - \arctan2\left(l_{d3}\sin\theta_{ds3}, l_{d2} + l_{d3}\cos\theta_{ds3}\right) \tag{27}$$

Here, to avoid multiple solutions, we preselect the human configuration: '*knee front*'.

It is worth to notice from the equations above, that when $-1 < D < 1$, at least one geometrically feasible solution exists. In other cases, the target pose is not reachable. The existence of a solution does not mean that the target pose is inside the humanoid's workspace, because the condition above does not consider the mechanical limits of joints.

Finally, to obtain the last joint coordinate, θ_{ds4}, we must define an auxiliary vector $\boldsymbol{L_4}$ that points from J_{ds4} to J_{ds5}, representing the link L_{ds4} (Figure 14).

Once this is done, we must compute the angle β between vectors $\boldsymbol{V_2}$ and $\boldsymbol{L_4}$ (Figure 15):

$$\beta = \cos^{-1}\frac{\boldsymbol{V_2}\cdot\boldsymbol{L_4}}{\|\boldsymbol{V_2}\|\, l_{d4}} \tag{28}$$

Nevertheless, Equation (28) is also incomplete. We need to provide its sign based on which vector is at the front with respect to the movement direction of the robot. One way to know the sign of the equation, is through the direction of the cross product between vectors $\boldsymbol{V_2}$ and $\boldsymbol{L_4}$, specifically the z component of the cross product. That is,

$$\beta = \mathrm{sgn}\left(\hat{\boldsymbol{k}}\cdot(\boldsymbol{V_2}\times\boldsymbol{L_4})\right)\cos^{-1}\frac{\boldsymbol{V_2}\cdot\boldsymbol{L_4}}{\|\boldsymbol{V_2}\|\, l_{d4}} \tag{29}$$

Another auxiliar angle we must define is α (Figure 16), obtained indirectly by the Law of Sines.

That is because this angle will sometimes be greater than $90°$, a case that is outside the arcsine's image. So α is calculated as follows:

$$\alpha = \pi - \left(\sin^{-1}\frac{l_{d3}\sin(\pi+\theta_{ds3})}{\|\boldsymbol{V_2}\|} + (\pi+\theta_{ds3})\right) \tag{30}$$

Once this is done, θ_{ds4} is calculated as:

$$\theta_{ds4} = \alpha + \beta \tag{31}$$

V. Kinematic Simulator Development

In order to validate the algorithms proposed in this paper, a kinematic simulator was developed (Figure 17). This simulator uses a VRML (*Virtual Reality Modeling Language*) model of the humanoid, obtained by exporting the corresponding CAD model generated in Pro/Engineer and structuring it according to the kinematic diagram shown

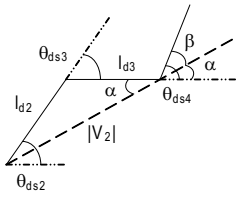

Fig. 16. θ_{ds4} related to the angles α and β

in Figure 2 [11]. This VRML model is embedded into a Visual Basic application that drives the model by sending the appropiate joint coordinates. These coordinates are directly specified by the user (Figure 18) or calculated from a desired pose, specified in task space (Figure 19). Once these joint coordinates are sent to the VRML model, it evolves from the actual pose to the final one by means of an interpolator, considering the body or any foot as the base for this model by means of an appropiate selector. Using the body or any foot as the base, allows to simulate the situation when the humanoid is hung by its body, or when it is supported by one foot (or both), as required in the corresponding walking stages. It is also possible to track the actual joint coordinates as the model evolves by using appropiate sensors, as well as the actual pose of the feet and body. These can be used to verify that both kinematic algorithms yield the desired results.

VI. Conclusions

The kinematic models developed in this paper were validated with the aid of the kinematic simulator just described. By giving different sets of joint coordinates, the humanoid always reached a pose that coincides with that calculated using (7). In addition, this simulator was given different target poses in task space too. When a desired pose was inside humanoid's workspace, this target pose was successfully reached, as seen in the corresponding sensor indicators. Otherwise, a different pose is achieved if $-1 < D < 1$ and at least one required joint is outside its range of motion. If $-1 < D < 1$ is not satisfied, the simulator simply ignores the command.

It is worth to mention that the IK algorithm proposed is computationally efficient. It only takes a fixed time of $16ms$ to obtain the desired solution. That is because it is an analytical solution instead of an approximate solution based on numerical methods that needs several iterations to accomplish the desired task. This is important because, in the near future, this algorithm can be used in real time to generate a parametrized walking pattern.

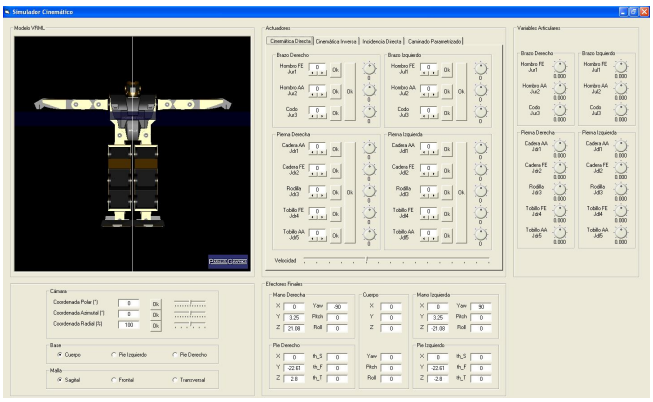

Fig. 17. Kinematic Simulator

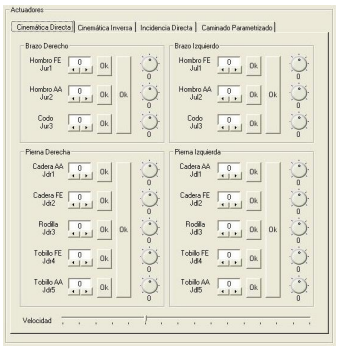

Fig. 18. Direct Kinematics Controls

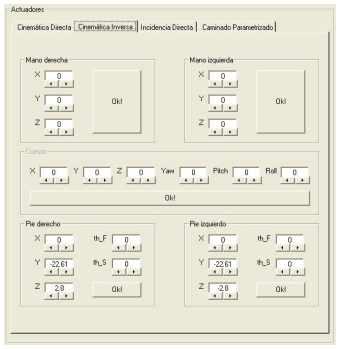

Fig. 19. Inverse Kinematics Controls

References

[1] Gaurav Tevatia and Stefan Schaal. Inverse kinematics for humanoid robots. *IEEE International Conference on Robotics and Automation*, 2000.

[2] Samuel R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Technical report, University of California, 2004.

[3] Aydemir Memişoğlu, Uğur Güdükbay, and Bülent Özgüç. Motion control for realistic walking behavior using inverse kinematics. *3DTV Conference*, 2007.

[4] J.J. Craig. *Introduction to Robotics*. Addison-Wesley, 1986.

[5] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[6] Edmond S.L. Ho, Taku Komura, and Rynson W.H. Lau. Computing inverse kinematics with linear programming. *ASME VRST*, 2005.

[7] Xuyang Wang, Tiansheng Lu, and Peiyan Zhang. State generation method for humanoid motion planning based on genetic algorithm. *Journal of Humanoids*, 2008.

[8] Javier De Lope, Rafaela Gonzlez-Careaga, Telmo Zarraonandia, and Dario Maravall. Inverse kinematics for humanoid robots using artificial neural networks. *Computer Aided Systems Theory (Eurocast 2003)*, 2003.

[9] Robert L. Williams II. Inverse kinematics and singularities of manipulators with offset wrist. *IASTED International Journal of Robotics and Automation*, 1999.

[10] Mark W. Spong and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley & Sons, 1989.

[11] J.M. Ibarra Zannatha and Rafael Cisneros Limón. Modelado y simulación de un humanoide. *7a. Conferencia Iberoamericana en Sistemas, Cibernética e Informática*, 2008.