Complete Analytical Inverse Kinematics for NAO

Nikos Kofinas, Emmanouil Orfanoudakis, and Michail G. Lagoudakis
Intelligent Systems Laboratory, Department of Electronic and Computer Engineering
Technical University of Crete, Chania, Crete 73100, Greece
Email: {nikofinas, vosk, lagoudakis}@intelligence.tuc.gr

***Abstract*—The design of complex dynamic motions for humanoid robots is achievable only through the use of robot kinematics. In this paper, we study the problems of forward and inverse kinematics for the Aldebaran NAO humanoid robot and present a complete, exact, analytical solution to both problems, including a software library implementation for real-time onboard execution. The forward kinematics allow NAO developers to map any configuration of the robot from its own joint space to the three-dimensional physical space, whereas the inverse kinematics provide closed-form solutions to finding joint configurations that drive the end effectors of the robot to desired target positions in the three-dimensional physical space. The proposed solution was made feasible through a decomposition into five independent problems (head, two arms, two legs), the use of the Denavit-Hartenberg method, and the analytical solution of a non-linear system of equations. The main advantage of the proposed inverse kinematics solution compared to existing approaches is its accuracy, its efficiency, and the elimination of singularities. In addition, we suggest a generic guideline for solving the inverse kinematics problem for other humanoid robots. The implemented, freely-available, NAO kinematics library, which additionally offers center-of-mass calculations, is demonstrated in two motion design tasks: basic center-of-mass balancing and pointing to the ball.**

I. Introduction

Articulated robots with multiple degrees of freedom, such as humanoid robots, have become popular research platforms in robotics and artificial intelligence. Our work focuses on autonomous humanoid platforms with multiple manipulators capable of performing complex motions, such as balancing, walking, and kicking. These skills are required in the Standard Platform League of the RoboCup robot soccer competition [1], in which all teams compete using the Aldebaran NAO humanoid robot [2], which is our target robot platform.

The design of complex dynamic motions is achievable only through the use of robot kinematics, which is an application of geometry to the study of arbitrary robotic chains. However, past work [3]–[5] has not fully solved the inverse kinematics problem for the NAO robot, since it focuses exclusively on the robot legs. Furthermore, the widely-known analytical solution [3] for the inverse kinematics of the legs is purely geometric and cannot be generalized to other kinematic chains. Also, existing numerical solutions [5] are inherently prone to singularities and, therefore, lack in robustness.

In this paper, we present a complete and exact analytical forward and inverse kinematics solution for all limbs of the Aldebaran NAO humanoid robot, using the established Denavit–Hartenberg convention [6], [7] for revolute joints. The main advantage of the proposed solution is its accuracy, its efficiency, and the elimination of singularities. In addition, we contribute an implementation of the proposed NAO kinematics as a freely-available software library[1] for real-time execution on the robot. This work enables NAO software developers to make transformations between configurations in the joint space and points in the three-dimensional physical space and vice-versa, on-board in just microseconds, as the library is designed for high-performance real-time execution on the limited embedded platform of the robot. The implemented NAO kinematics library, which additionally offers center-of-mass calculations, is demonstrated in two tasks[2]: basic center-of-mass balancing and pointing to the ball. The library has been integrated into the software architecture of our RoboCup team *Kouretes* [www.kouretes.gr] and is currently being used in various motion design problems, such as dynamic balancing, trajectory following, dynamic kicking, and omnidirectional walking. Extrapolating from our work on the NAO, we also present some guidelines for finding analytical solutions to the inverse kinematics problem for any humanoid with revolute joints of up to 6 degrees of freedom (DOF) per manipulator.

II. Background

A. The Aldebaran NAO Humanoid Robot

NAO (v3.3) is a 58cm, 5kg humanoid robot (Figure 1). The NAO robot carries a fully capable computer on-board with an x86 AMD Geode processor at 500 MHz, 256 MB SDRAM, and 2 GB flash memory running an Embedded Linux distribution. It is powered by a 6-cell Lithium-Ion battery which provides about 30 minutes of continuous operation and communicates with remote computers via an IEEE 802.11g wireless or a wired Ethernet link. NAO RoboCup edition has 21 degrees of freedom; 2 in the head, 4 in each arm, 5 in each leg and 1 in the pelvis (there are two pelvis joints which are coupled together on one servo and cannot move independently). All joints are position-controlled, using closed-loop PID controllers and encoders. It also features a variety of sensors: an Inertial Measurement Unit (IMU) in the torso, Force Sensitive Resistors (FSR) on each foot, ultrasonic range sensors on the chest, and two VGA cameras on the head.

B. Transformation Formalism

The translation and orientation of a joint j with respect to an adjacent joint i in the three-dimensional space can be fully

[1]Library download link: www.github.com/kouretes/NAOKinematics
[2]Video download link: www.intelligence.tuc.gr/kouretes/NAOKinematics

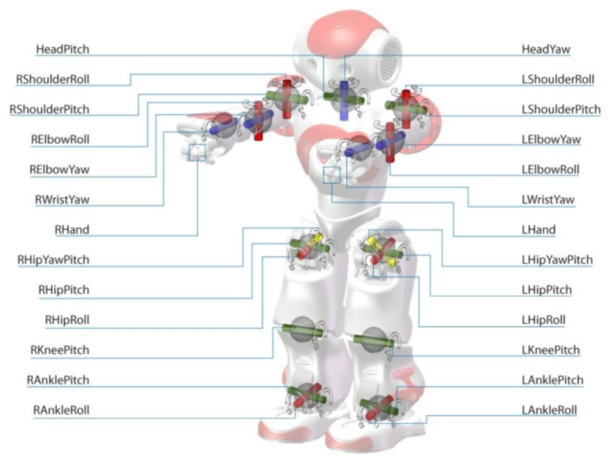

Fig. 1. NAO v3.3 kinematic chains and joints (Academics Edition, 25 DOF)

described using a 4×4 *(affine) transformation matrix* $\mathbf{T}_i^j$:

$$\mathbf{T}_i^j = \begin{bmatrix} \mathbf{X} & \bar{y} \\ \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix} & 1 \end{bmatrix} \tag{1}$$

where $\mathbf{X} \in \Re^{3\times 3}$ and $\bar{y} \in \Re^3$. A transformation matrix $\mathbf{T}_i^j$ provides the translation ($\bar{y}$) and orientation (contained in $\mathbf{X}$) of a coordinate system j with respect to coordinate system i. A transformation matrix is invertible, if and only if $\mathbf{X}$ is invertible, and is formed as:

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{X}^{-1} & -\mathbf{X}^{-1}\bar{y} \\ \begin{bmatrix} 0 & \cdots & 0 \end{bmatrix} & 1 \end{bmatrix}$$

Given a robotic manipulator of N joints, an equal number of left-handed Cartesian coordinate systems (frames) are established, each affixed to the previous one, and the one-to-one transformation between them forms a transformation matrix.

For convenience, we enumerate joint frames starting from an established base frame, typically a fixed point on the robot's body. A point $\bar{p}_j = \begin{bmatrix} p_x & p_y & p_z & 1 \end{bmatrix}^\top$ described in frame j can be transformed to a point $\bar{p}_i$ in another frame i by cascading the transformations for all intermediate frames:

$$\mathbf{T}_i^j = \mathbf{T}_i^{i+1}\mathbf{T}_{i+1}^{i+2}\cdots\mathbf{T}_{j-1}^j \quad \text{and} \quad \bar{p}_i = \mathbf{T}_i^j \bar{p}_j$$

For the needs of forward and inverse kinematics, we utilize translations and rotations. A *translation transformation* has $\mathbf{X} = \mathbf{I}_3$ (the identity matrix) and the desired offset as $\bar{y}$ in Eq. 1. We denote a parametric translation matrix for $\bar{y} = \bar{t}$ as $\mathbf{A}(\bar{t})$. It can be trivially shown that $\mathbf{A}^{-1}(\bar{t}) = \mathbf{A}(-\bar{t})$ and $\mathbf{A}(\bar{w}+\bar{z}) = \mathbf{A}(\bar{w})\mathbf{A}(\bar{z})$. A *rotation transformation* has $\bar{y} = \bar{0}$ (no translation) and $\mathbf{X}$ in Eq. 1 is an arbitrary rotation matrix $\mathbf{R}$ $\left(\mathbf{R}^{-1} = \mathbf{R}^\top \text{ and } \det(\mathbf{R}) = 1\right)$. We denote the elementary rotation matrices about the x, y, z axes as $\mathbf{R}_{\text{axis}}(\text{angle})$. All rigid body transformations related to kinematics consist of cascaded elementary transformations (translations and rotations) and, therefore, are always invertible.

C. Denavit–Hartenberg Convention

The established formalism for describing transformations between two frames adjacent to a joint is the Denavit-Hartenberg (DH) parameters: $\mathtt{a}$, α, d, and θ. For the NAO, these parameters are provided by the manufacturer. The current angle (state) of the joint is θ. Given the parameters of some joint j, the DH transformation that describes the translation and orientation of the reference frame of joint j with respect to the reference frame of the previous joint $j-1$ is:

$$\mathbf{T}_{j-1}^j = \mathbf{R}_x(\alpha_j)\mathbf{A}\left(\begin{bmatrix} \mathtt{a}_j & 0 & 0 \end{bmatrix}^\top\right)\mathbf{R}_z(\theta_j)\mathbf{A}\left(\begin{bmatrix} 0 & 0 & d_j \end{bmatrix}^\top\right)$$

Being a product of invertible matrices, a DH transformation matrix is always invertible.

D. Transformation Decomposition

An arbitrary transformation matrix can be decomposed as a "translation after rotation" pair:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \bar{y} \\ \bar{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \bar{y} \\ \bar{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \bar{0} \\ \bar{0}^\top & 1 \end{bmatrix}$$

Using the *Yaw-Pitch-Roll* convention, any rotation matrix $\mathbf{R}$ decomposes into a product of the three elementary rotations:

$$\mathbf{R} = \mathbf{R}_z(a_z)\mathbf{R}_y(a_y)\mathbf{R}_x(a_x)$$

The orientation vector $\begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^\top$ can be extracted analytically from any rotation matrix. Therefore, any *position* in the three-dimensional space, described by the six values of a translation vector $\begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^\top$ and an orientation vector $\begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^\top$, defines a unique transformation matrix.

III. NAO Forward Kinematics Solution

Taking the torso frame of the NAO robot as the base frame, the forward kinematic equations for the five kinematic chains of NAO (RoboCup Edition) are the following:

$$\mathbf{T}_{\text{Base}}^{\text{Head}} = \mathbf{A}_{\text{Base}}^0\mathbf{T}_0^1\mathbf{T}_1^2\mathbf{R}_x(\tfrac{\pi}{2})\mathbf{R}_y(\tfrac{\pi}{2})\mathbf{A}_2^{\text{Head}} \tag{2}$$
$$\mathbf{T}_{\text{Base}}^{\text{LHand}} = \mathbf{A}_{\text{Base}}^0\mathbf{T}_0^1\mathbf{T}_1^2\mathbf{T}_2^3\mathbf{T}_3^4\mathbf{R}_z(\tfrac{\pi}{2})\mathbf{A}_4^{\text{LHand}} \tag{3}$$
$$\mathbf{T}_{\text{Base}}^{\text{RHand}} = \mathbf{A}_{\text{Base}}^0\mathbf{T}_0^1\mathbf{T}_1^2\mathbf{T}_2^3\mathbf{T}_3^4\mathbf{R}_z(\tfrac{\pi}{2})\mathbf{A}_4^{\text{RHand}}\mathbf{R}_z(-\pi) \tag{4}$$
$$\mathbf{T}_{\text{Base}}^{\text{LFoot}} = \mathbf{A}_{\text{Base}}^0\mathbf{T}_0^1\mathbf{T}_1^2\mathbf{T}_2^3\mathbf{T}_3^4\mathbf{T}_4^5\mathbf{T}_5^6\mathbf{R}_z(\pi)\mathbf{R}_y(-\tfrac{\pi}{2})\mathbf{A}_6^{\text{LFoot}} \tag{5}$$
$$\mathbf{T}_{\text{Base}}^{\text{RFoot}} = \mathbf{A}_{\text{Base}}^0\mathbf{T}_0^1\mathbf{T}_1^2\mathbf{T}_2^3\mathbf{T}_3^4\mathbf{T}_4^5\mathbf{T}_5^6\mathbf{R}_z(\pi)\mathbf{R}_y(-\tfrac{\pi}{2})\mathbf{A}_6^{\text{RFoot}} \tag{6}$$

where each $\mathbf{T}_j^i$ in the equations above is the DH transformation matrix between joints i and j in the corresponding chain and the $\mathbf{A}$'s are translation matrices defined by the specifications of the robot (lengths of limbs) [5].

Should we need to extract the position of some manipulator b with respect to another a (e.g. head with respect to left leg), we can construct two such chains $\mathbf{T}_c^a$, $\mathbf{T}_c^b$ from a common point c (e.g. Base) and combine them as $\mathbf{T}_a^b = \left(\mathbf{T}_c^a\right)^{-1}\mathbf{T}_c^b$.

IV. Solving the Inverse Kinematics Problem

Precise control of manipulators and effectors can be achieved by solving the inverse kinematics problem, whereby the values θ_i of the angles of various joints must be determined to place the manipulator at a specific target position (translation and/or orientation). The solution of the inverse problem is robot-specific and generally under/over-determined kinematic chains exist. Iterative numerical solutions may converge to a solution, but, in general, suffer from singularities and poor performance [8]. On the other hand, analytical solutions are fast and exact, but require significant effort in extracting them.

A. Inverse Kinematics Methodology

The following seven steps were taken in order to find a complete solution for the inverse kinematics problem for all the kinematic chains of the NAO humanoid robot.

1) Construct the numeric transformation: Given a desired target position, denoted by an orientation vector $\bar{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^\top$ and a translation vector $\bar{p} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^\top$, it is easy to reconstruct the target transformation matrix:

$$\mathbf{T} = \mathbf{A}(\bar{p})\mathbf{R}_z(a_z)\mathbf{R}_y(a_y)\mathbf{R}_x(a_x)$$

2) Construct the symbolic transformation: Setting all θ parameters as unknowns in the forward kinematics solution of the target kinematic chain yields a symbolic matrix:

$$\mathbf{T}^0_{\text{base}}\mathbf{T}^j_0(\theta_0, \ldots, \theta_j)\mathbf{T}^{\text{end}}_j$$

3) Form a non-linear system: By equating the above matrices, a non-linear system is formed, since the unknown θ's appear in transcendental trigonometric forms. Now, the problem is to find values for the θ's from 12 equations (the upper 3×4 block) of which only up to six are independent.

$$\mathbf{T} = \mathbf{T}^0_{\text{base}}\mathbf{T}^j_0(\theta_0, \ldots, \theta_j)\mathbf{T}^{\text{end}}_j$$

4) Manipulate both sides: The chain can be simplified by eliminating known terms. Such terms (e.g. the base and the end transformations) can be removed by multiplying both sides of the system with the appropriate inverse matrix:

$$\left(\mathbf{T}^0_{\text{base}}\right)^{-1}\mathbf{T}\left(\mathbf{T}^{\text{end}}_j\right)^{-1} = \mathbf{T}^j_0(\theta_0, \ldots, \theta_j)$$

As soon as we find a solution for some θ_i, we can remove in a similar way the corresponding joint i from the chain, because the corresponding DH transformation matrix is now known; this can occur if and only if this joint is the first or the last in the kinematic chain.

Another way to manipulate the chain is to induce arbitrary (known) constant transformations at the beginning or the end of the chain, aiming at simplifying the non-linear system.

$$\mathbf{T}_c(\mathbf{T}^0_{\text{base}})^{-1}\mathbf{T}(\mathbf{T}^{\text{end}}_j)^{-1} = \mathbf{T}_c\mathbf{T}^j_0(\theta_0, \ldots, \theta_j)$$

In some kinematic chains we can decouple the orientation and translation sub-problems. Quite often the target translation vector can be expressed as a function of fewer joints in the analytical equation of the kinematic chain or in the analytical equation of the reverse kinematic chain.

5) Use geometry and trigonometry: It is possible to form a closed-form solution for some θ_j using a geometric model of the chain. For chains with up to two links (non-zero a and d parameters) or "arm and wrist" chains commonly found in humanoids, a geometric approach can easily determine the values for the joints that lie between the links. These joints can be modeled as an angle of the triangle formed by the links, so the value of the joint can be obtained using trigonometry.

The kinematic leg chain of the NAO robot, for example, has such a joint. Figure **??** shows the triangle formed by the right leg, the lower leg, and the line that connects the base with the target point. Upper and lower leg lengths are known and the third side of the triangle can be computed using the Euclidean distance between the base of the chain and the end of the chain. The law of cosines, yields a set of complementary closed-form solutions for the angle θ.

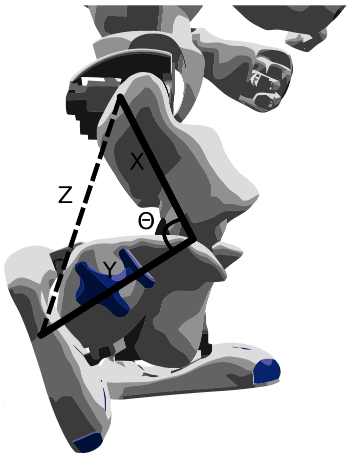

Fig. 2. Triangle formed by the robot leg

6) Solve the non-linear system: The resulting equations are combinations of $\sin\theta_i$ and $\cos\theta_i$, thus, the closed-form solution of these equations must utilize the inverse trigonometric functions ($acos$, $asin$). The transcendental nature of the $acos$ and $asin$ trigonometric functions has the inherent problem of producing multiple solutions in $[-\pi, \pi]$. Without any restrictions on the valid range of a joint, we must examine all candidate solutions for each joint and their combinations for validity. To avoid this multiplicity, solutions that rely on $atan$ and $acot$ are preferred, but forming them might not be possible for a particular chain.

7) Validate through forward kinematics: Generally, there are multiple candidate solutions for the joint values, due to the existence of complementary and/or supplementary angles. A validation step is taken to discard invalid candidates. This validation is performed by feeding each candidate solution to the forward kinematics of the chain and checking whether the resulting position matches precisely the target position. Choosing among the valid solutions, if more than one, can be addressed independently of kinematics.

B. Applicability

The methodology presented above offers a generic guideline for solving the inverse kinematics problem on typical humanoid robot kinematic chains that have the generic two-link configuration (found in both the arms and legs). More specifically, the kinematic chains must have up to five joints or six joints with three consecutive ones having intersecting axes [9], [10] to expect a possible solution.

V. NAO Inverse Kinematics Solution

Using the methodology presented above, we find the inverse kinematics solution for all five kinematic chains of NAO (RoboCup Edition): head (2 joints), left arm (4 joints), right arm, left leg (6 joints), and right leg. The left chains are almost identical to the right ones, thus the solutions are similar. Due

$$\theta_2 = asin\left(\frac{-p_z + l_3}{\sqrt{{l_1}^2 + {l_2}^2}}\right) - atan\left(\frac{l_1}{l_2}\right) + \frac{\pi}{2}$$

$$\theta_2 = \pi - asin\left(\frac{-p_z + l_3}{\sqrt{{l_1}^2 + {l_2}^2}}\right) - atan\left(\frac{l_1}{l_2}\right) + \frac{\pi}{2}$$

$$\theta_1 = \pm acos\left(\frac{p_x}{l_2 \cos\left(\theta_2 - \frac{\pi}{2}\right) - l_1 \sin\left(\theta_2 - \frac{\pi}{2}\right)}\right)$$

provided a target translation (p_x, p_y, p_z), or

$$\theta_1 = a_z \quad \theta_2 = a_y$$

provided a target orientation (a_x, a_y, a_z)

Fig. 3. Head Inverse Kinematics Solution

to space restrictions, the solutions for the the left part are presented in detail, whereas the ones for the right part are abbreviated. Full details may be found in a longer technical report [11] .

A. Inverse Kinematics for the Head Chain

The head chain consists of only two joints (HeadYaw, HeadPitch—in this order), therefore we can solve for either the translation ($\bar{p}$) or the orientation ($\bar{a}$) of the target position to obtain a solution. In the latter case, we can achieve the desired target orientation simply by setting the HeadYaw and HeadPitch joints to a_z and a_y respectively, and assume $a_x = 0$. In the former case, we construct the symbolic matrix through the forward kinematics solution (Eq. 2). Now, we can equate the translation part from the symbolic matrix with $\bar{p}$ and from these equations we can easily find the desired θ values. Figure 3 shows the resulting analytical solution, in which l_1 and l_2 are the x and the y part of the end translation and l_3 is the z part of the base translation.

B. Inverse Kinematics for the Left Arm Chain

The left arm chain consists of four joints (LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll—in this order). Following our methodology, the first three steps are straightforward given the forward kinematics solution (Eq. 3). Next, using the fourth step, we manipulate our chain to remove known translations and rotations. Then we invert the chain because it is easier to extract joint values from the new translation part. From the translation part of the inverted transformation matrix we extract the values of θ_3 and θ_4. After that we manipulate again our chain to remome the (now known) DH-transformation for the last two joints. Finally, we validate all candidate solutions through the forward kinematics validation step. Figure 4 shows the resulting analytical solution, in which $\bar{s}$ is the base translation vector, l_1 is the offset of the elbow, l_2 is the length of the upper arm, and $\mathbf{T}_{(i,j)}$ is the (i, j) element of matrix $\mathbf{T}$.

C. Inverse Kinematics for the Right Arm Chain

The right arm chain is almost identical to the left arm chain. The only difference is in the forward kinematics solution, since the a parameter for θ_3 has a minus sign and some joint have mirrored values. Besides this difference, all other steps have similar results. The analytical solution is the one shown in Figure 4 with the following differences: (a) there is a minus sign in front of all the instances of l_1, (b) for all the "roll" joints we must give new limits.

$$\mathbf{T}' = \left(\mathbf{A}^0_{\text{Base}}\right)^{-1} \mathbf{T} \left(\mathbf{A}^{\text{End}}_4\right)^{-1} \left(\mathbf{R}_z(\tfrac{\pi}{2})\right)^{-1}$$

$$\mathbf{T}'' = \left(\mathbf{T}'\right)^{-1}$$

$$\theta_3 = \begin{cases} asin\left(\frac{\mathbf{T}''_{(3,4)}}{l_1}\right) \\ \pi - asin\left(\frac{\mathbf{T}''_{(3,4)}}{l_1}\right) \end{cases}$$

$$\theta_4 = \pm acos\left(\frac{l_3 \mathbf{T}''_{(2,4)} - l_2 \mathbf{T}''_{(1,4)} \cos\theta_3}{l_2^2 + l_1^2 \cos^2\theta_3}\right)$$

$$\mathbf{T}''' = \mathbf{T}' \left(\mathbf{T}^3_2\right)^{-1} \left(\mathbf{T}^4_3\right)^{-1}$$

$$\theta_2 = atan\left(\frac{\mathbf{T}'''_{(2,1)}}{\mathbf{T}'''_{(2,2)}}\right) - \frac{\pi}{2}$$

$$\theta_1 = atan\left(\frac{\mathbf{T}'''_{(1,3)}}{\mathbf{T}'''_{(3,3)}}\right)$$

Fig. 4. Left Arm Inverse Kinematics Solution

$$\mathbf{T}' = \left(\mathbf{R}_x(\tfrac{\pi}{4}) \left(\left(\mathbf{A}^0_{\text{Base}}\right)^{-1} \mathbf{T} \left(\mathbf{A}^{\text{End}}_6\right)^{-1}\right)\right)^{-1}$$

$$\theta_4 = \pm\left(\pi - acos\left(\frac{{l_1}^2 + {l_2}^2 - \|\bar{0} - \bar{p}\|_2}{2 l_1 l_2}\right)\right)$$

$$\theta_6 = \begin{cases} atan\left(\frac{\mathbf{T}'_{(2,4)}}{\mathbf{T}'_{(3,4)}}\right) & \text{if } (l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)) \neq 0 \\ undefined & \text{if } (l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)) = 0 \end{cases}$$

$$\mathbf{T}'' = \left(\left(\mathbf{T}'\right)^{-1} \left(\mathbf{T}^6_5 \mathbf{R}_z(\pi) \mathbf{R}_y(-\tfrac{\pi}{2})\right)^{-1}\right)^{-1}$$

$$\theta_5 = asin\left(-\frac{\mathbf{T}''_{(2,4)} (l_2 + l_1 \cos\theta_4) + l_1 \mathbf{T}''_{(1,4)} \sin\theta_4}{{l_1}^2 \sin^2\theta_4 + (l_2 + l_1 \cos\theta_4)^2}\right)$$

$$\theta_5 = \pi - asin\left(-\frac{\mathbf{T}''_{(2,4)} (l_2 + l_1 \cos\theta_4) + l_1 \mathbf{T}''_{(1,4)} \sin\theta_4}{{l_1}^2 \sin^2\theta_4 + (l_2 + l_1 \cos\theta_4)^2}\right)$$

$$\mathbf{T}''' = \left(\mathbf{T}''\right)^{-1} \left(\mathbf{T}^4_3 \mathbf{T}^5_4\right)^{-1}$$

$$\theta_2 = \pm acos\left(\mathbf{T}'''_{(2,3)}\right) - \frac{\pi}{4}$$

$$\theta_3 = asin\left(\frac{\mathbf{T}'''_{(2,2)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right) \qquad \theta_3 = \pi - asin\left(\frac{\mathbf{T}'''_{(2,2)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right)$$

$$\theta_1 = \pm acos\left(\frac{\mathbf{T}'''_{(1,3)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right) + \frac{\pi}{2}$$

Fig. 5. Left Leg Inverse Kinematics Solution

D. Inverse Kinematics for the Left Leg Chain

The kinematic chain of the left leg has six joints (LHipYawPitch, LHipRoll, LHipPitch, LKneePitch, LAnklePitch, LAnkleRoll—in this order), but since the first three joints have intersecting axes, the problem is possibly solvable [9], [10]. We construct both the numerical and symbolic parts of the system with the help of forward kinematics (Eq. 5). Following the fourth step, to make the problem easier, we remove the known translations from the kinematic chain. Then, to simplify the solution we induce a $\mathbf{R}_x(\frac{\pi}{4})$ transformation at the start of the chain. In effect, we transform the first joint from a yaw-pitch joint to a yaw joint, which is simpler to handle. Close examination of the resulting kinematic chain reveals that the first four joints are responsible for the translation part and all six joints are responsible for the orientation part. It would be convenient, if only three joints were affecting the translation of the end effector, because in that case we could extract these joints just from the translation part. Thus, we invert the transformation matrix to form the reverse chain. Now, only three joints (LAnkleRoll, LAnklePitch, LKneePitch) affect the translation. We can now find θ_4 using the fifth step as we mentioned above. We focus on the triangle formed by the UpperLeg, LowerLeg, and the line connecting the base to the target point. Next, the θ_5 and θ_6 angles can be extracted from the translation part. The solution we found for θ_6 has some undefined points, because the denominator of the result may become zero. These undefined points are discussed in Section VI.

After we calculate θ_4, θ_6, and θ_5 from the translation part, we can go back to step four and remove the, now, known DH transformation matrices from the chain. The resulting kinematic chain consists of only three joints, which control only the orientation. It is easy to extract the remaining joint values from the nine equations of the rotation block. Figure 5 shows the resulting analytical solution, in which l_1 is the length of the upper leg and l_2 is the length of the lower leg.

E. Inverse Kinematics for the Right Leg Chain

The right leg chain is identical to the left leg chain. The only difference is the DH parameter α of the first joint. Thus, we must multiply instead with $\mathbf{R}_x(-\frac{\pi}{4})$. Otherwise, the solution is exactly the same as the solution shown in Figure 5 with all instances of $(\theta_2 + \frac{\pi}{4})$ changed to $(\theta_2 - \frac{\pi}{4})$.

VI. Implementation

Having completed all kinematics in analytical form, we created `NAOKinematics`, a software library for real-time, onboard execution of NAO kinematics in `C++`. Given that `C++` offers no library for optimized real-time matrix operations, we relied on our linear algebra framework KMat [12] for such operations. A Matlab version of the library is also available for other applications. Our library includes five functions for calculating the forward kinematics for each chain, given the corresponding joint values. It also includes five functions, whose input is the desired target position and output is a set of solutions, for all the joints of a specified chain. The library also includes a function for calculating the center of mass of the robot given a set of values for all joints.

As mentioned before, there are a few target positions for the legs which lead to an infinity of solutions for the AnkleRoll joint, when the KneePitch and AnklePitch joints take specific values and essentially cancel the effect of AnkleRoll on the translation of the reverse chain. Figure 6 shows one of these problematic configurations. The locus of these configurations is a line in the configuration space (Figure 7). To verify that in practice the robot never reaches any of these configurations, we let the robot perform the entire range of motions available to it during operation in a RoboCup field (walk, kicks, stand-up, etc.) and plotted these motions alongside the problematic locus. The results are shown in Figure 7. It is clear that no motion brought the robot to these configurations. In practice, it is rather unlikely that anyone will consistently give target positions that drive the joints in that area, during regular use.

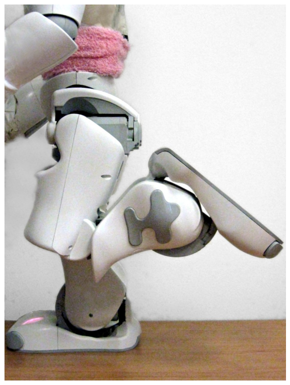

Fig. 6. An instance of the problematic leg configurations

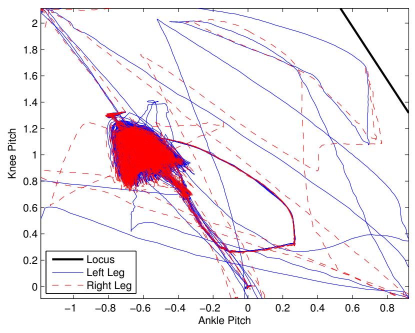

Fig. 7. Trajectories of motion in a subspace of the leg joints

VII. Results

A. Real-Time Performance

One of the goals of this work was to implement a software library for real-time kinematics computations on the robot. We measured the performance of our implementation for each of the functions we offer. Table I shows average execution times.

B. Demonstration I: Basic CoM Balancing

In this demonstration, we seek to implement a very basic balancing method. In particular, we want to make NAO move

TABLE I
On-board execution times of the NAOKinematics library

Kinematics Function	**Time** (μs)
Forward Kinematics for Head	54.28
Forward Kinematics for Arm	66.72
Forward Kinematics for Leg	80.88
Inverse Kinematics for Head	70.79
Inverse Kinematics for Arm	170.55
Inverse Kinematics for Leg	185.29
Calculation of the Center of Mass	394.55

one of its feet to the point of the projection of the Center of Mass (CoM) on the floor. First, we calculate the translation of the CoM relatively to the torso frame using forward kinematics. The problem is that the x-y plane of the torso frame is rarely parallel to the floor. Thus, we read off the inertial unit of the robot the current rotation (angleX and angleY) of the torso plane. Now, we can calculate the translation of the CoM relatively to the rotated torso:

$$\mathbf{T}_{\text{rotated}} = \mathbf{R}_y(\text{angleY})\mathbf{R}_x(\text{angleX})\mathbf{A}(\text{CoM})$$

Then, we assign a custom value to p_z in $\mathbf{T}_{(4,3)}$, which represents the desired torso height from the floor and that yields $\mathbf{T}'_{\text{rotated}}$. Now we must rotate back to the torso frame:

$$\mathbf{T}_{\text{final}} = \big(\mathbf{R}_y(\text{angleY})\mathbf{R}_x(\text{angleX})\big)^{-1}\mathbf{T}'_{\text{rotated}}$$

Finally, we extract p_x, p_y, and p_z from $\mathbf{T}_{\text{final}}$ and we set $\begin{bmatrix}p_x & p_y & p_z\end{bmatrix}^\top$ as the target translation for inverse kinematics. The target orientation is set to $\begin{bmatrix}a_x & a_y & a_z\end{bmatrix}^\top = \begin{bmatrix}-\text{angleX} & -\text{angleY} & 0\end{bmatrix}^\top$, because we do not care about the rotation about the z-axis. Note that the foot is always parallel to the floor, excluding any hardware precision errors.

C. Demonstration II: Pointing to the Ball

In this demonstration, our goal is to make the NAO point to the ball with its stretched arms. Apart from the kinematics, to realize this task we employed our vision module [12] for ball recognition, along with the module that filters the belief of the robot about the ball location. Initially, NAO scans for the ball. When found, it points to it with the left, the right, or both arms, depending on where the ball is located (left, right, or front). The ball observation can be described as a two-dimensional translation (p_x, p_y) on the floor. We add the height of the torso (found through forward kinematics) as the third coordinate p_z to form the ball translation (p_x, p_y, p_z) in the three-dimensional space. We also set a_x to zero, because we are only rotating about the y-axis (up/down) and the z-axis (right/left). To find the other two orientations, we focus on the straight line that connects the location of the ball and the point of the ShoulderPitch joint relatively to the torso frame. The orientations a_y, a_z are the angles between this line and the corresponding axes. Additionally, the target point lies on this line at a distance equal to the length of the stretched arm from the ShoulderPitch joint. We run this procedure for both arms and obtain the solution(s) from inverse kinematics. If both solutions are returned, the robot raises both arms pointing to the ball. If only one solution is returned, the robot raises only one arm; the other arm cannot physically point to the ball.

VIII. Conclusion

In this paper, we presented a complete, exact, analytical solution for the problems of forward and inverse kinematics of the NAO robot. The main advantage of our solution is its accuracy, its efficiency, and the elimination of singularities. In addition, we contributed an implementation of the proposed NAO kinematics as a freely-available software library for real-time execution on the robot or for simulations.

Our approach to NAO kinematics is based on standard principled methods for studying robot kinematic chains. No complete analytical solution with full implementation for the NAO robot has been published before. The currently widely-known solution of team B-Human [3] applies only to the legs, is purely geometric, and cannot be generalized to other kinematic chains. In addition, their work has not studied the effect of the existing potentially problematic configurations. We have tried to implement the other published analytical solution for the legs by team MRL [4], but we were not able to reproduce their results. Finally, the numerical solution [5] offered by the manufacturer of the robot, Aldebaran Robotics, is a proprietary implementation, which unfortunately is inherently prone to singularities and, therefore, lacks in robustness. It should be noted that none of the two demonstrations we presented in this paper could be realized with the existing solutions and implementations of NAO kinematics.

Since kinematics is the base for several applications related to robot motion, we expect that our work will be useful not only to RoboCup SPL teams, but also to any NAO software developer. We believe that NAO developers can take advantage of our off-the-shelf NAO kinematics library to work on omni-directional walk algorithms, dynamic balancing methods, dynamic kick engines, etc. Our library can offer the basis for following dynamic trajectories in real time for walking and kicking or calculating the center of mass dynamically in real time for balancing.

Finally, our methodology offers a generic guideline for addressing the problem of inverse kinematics in humanoid robots. Apart from extending our work to the Academic Edition of NAO, which has four additional joints, one of our future goals is to apply the same methodology to robots similar to NAO, such as the Darwin-OP humanoid robot, which has a maximum of six degrees of freedom per kinematic chain.

References

[1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, "Robocup: A challenge problem for AI," *AI Magazine*, vol. 18, no. 1, pp. 73–85, 1997.

[2] D. Gouaillier and P. Blazevic, "A mechatronic platform, the Aldebaran Robotics humanoid robot," in *Proceedings of the 32nd IEEE Annual Conference on Industrial Electronics (IECON)*, 2006, pp. 4049–4053.

[3] C. Graf, A. Härtl, T. Röfer, and T. Laue, "A robust closed-loop gait for the Standard Platform League humanoid," in *Proceedings of the Fourth Workshop on Humanoid Soccer Robots*, 2009, pp. 30–37.

[4] M. G. Jadidi, E. Hashemi, M. A. Z. Harandi, and H. Sadjadian, "Kinematic modeling improvement and trajectory planning of the NAO biped robot," in *Proceedings of the 1st Joint International Conference on Multibody System Dynamics*, 2010.

[5] Aldebaran Robotics, "Nao documentation," 2012, only available online: www.aldebaran-robotics.com/documentation.

[6] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *ASME Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955.

[7] R. S. Hartenberg and J. Denavit, *Kinematic Synthesis of Linkages*. New York: McGraw-Hill, 1964.

[8] S. R. Buss, "Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least-squares methods," 2009, available at: www.math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf.

[9] D. Pieper and B. Roth, "The kinematics of manipulators under computer control," in *Proceedings of the 2nd International Congress on Theory of Machines and Mechanisms*, vol. 2, 1969, pp. 159–169.

[10] D. Pieper, “The kinematics of manipulators under computer control,” Ph.D. dissertation, Stanford University, U.S.A., 1968.

[11] N. Kofinas, “Forward and inverse kinematics for the NAO humanoid robot,” Diploma Thesis, Technical University of Crete, Greece, 2012, available at: www.intelligence.tuc.gr/lib/downloadfile.php?id=430.

[12] E. Orfanoudakis, “Reliable object recognition for the RoboCup domain,” Diploma Thesis, Technical University of Crete, Greece, 2011.