J Intell Robot Syst (2015) 77:251–264
DOI 10.1007/s10846-013-0015-4

Complete Analytical Forward and Inverse Kinematics for the NAO Humanoid Robot

Nikolaos Kofinas · Emmanouil Orfanoudakis · Michail G. Lagoudakis

Received: 5 July 2013 / Accepted: 26 December 2013 / Published online: 31 January 2014

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, the analytical solution of a non-linear system of equations, and the exploitation of body and joint symmetries. 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 and Jacobian inverse kinematics, is demonstrated in three motion design tasks: basic center-of-mass balancing, pointing to a moving ball, and human-guided balancing on two legs.

Keywords Robot kinematics · Humanoid robots · Aldebaran NAO robot

Mathematics Subject Classification (2010) 68T40 · 53A17

1 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 challenging tasks, such as the Standard Platform League of the RoboCup robot soccer competition [1], in which

N. Kofinas · E. Orfanoudakis · M. G. Lagoudakis (✉)
Intelligent Systems Laboratory, School of Electronic and Computer Engineering, Technical University of Crete, Chania, Crete 73100, Greece
e-mail: lagoudakis@intelligence.tuc.gr

N. Kofinas
e-mail: nikofinas@intelligence.tuc.gr

E. Orfanoudakis
e-mail: vosk@intelligence.tuc.gr

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. To our knowledge, no other complete inverse kinematics solutions for the NAO arms exist. Furthermore, the currently 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, may fail even if solutions exist 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 initialization dependencies and singularities commonly found in iterative numerical methods. In addition, we contribute an implementation of the proposed NAO kinematics as a freely-available, stand-alone software library[1] for real-time execution on the robot. Our 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 solution, which additionally offers center-of-mass calculations and Jacobian inverse kinematics, is demonstrated in three tasks accompanied by videos:[2] basic center-of-mass balancing, pointing to a moving ball, and human-guided balancing on two legs. 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: dynamic balancing, trajectory following, dynamic kicking, and omnidirectional walking. Extrapolating from our work on the NAO, we also present generic 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.

The remainder of the paper is organized as follows. In Section 2 we provide the required background: a description of our target robot platform, our transformation formalism, and a quick review of the Denavit-Hartenberg method and related transformation decompositions. Section 3 covers the forward kinematics solution for the NAO robot. Section 4 describes our methodology to finding analytical inverse kinematics solutions and, subsequently, Section 5 covers our inverse kinematics solution for the NAO robot. In Section 6 we give details about the implementation of the NAOKinematics library, while in Section 7 we demonstrate our approach in terms of real-time performance and application to three motion design tasks. We conclude with a discussion of related work and directions for future work in Section 8.

2 Background

2.1 The Aldebaran NAO Humanoid Robot

NAO H25 (v4.0) is a 58cm, 5kg humanoid robot (Fig. 1) manufactured by Aldebaran Robotics in Paris, France. The NAO robot carries a fully capable computer on-board with an x86 ATOM Z530 processor at 1.6 GHz, 1.0 GB of RAM, 2.0 GB of flash disk, and up to 8 GB of user flash memory running an Embedded Linux distribution. It is powered by a 6-cell Lithium-Ion battery which provides about 60 minutes of continuous operation and communicates with remote computers via an IEEE 802.11g wireless or a wired Ethernet link. NAO H25 has 25 degrees of freedom; 2 in the head, 6 in each arm, 5 in each leg and 1 in the pelvis (there are two pelvis joints which are coupled 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 960p@30fps cameras on the head.

[1] Library link: www.github.com/kouretes/NAOKinematics

[2] Video link: www.kouretes.gr/NAOKinematics

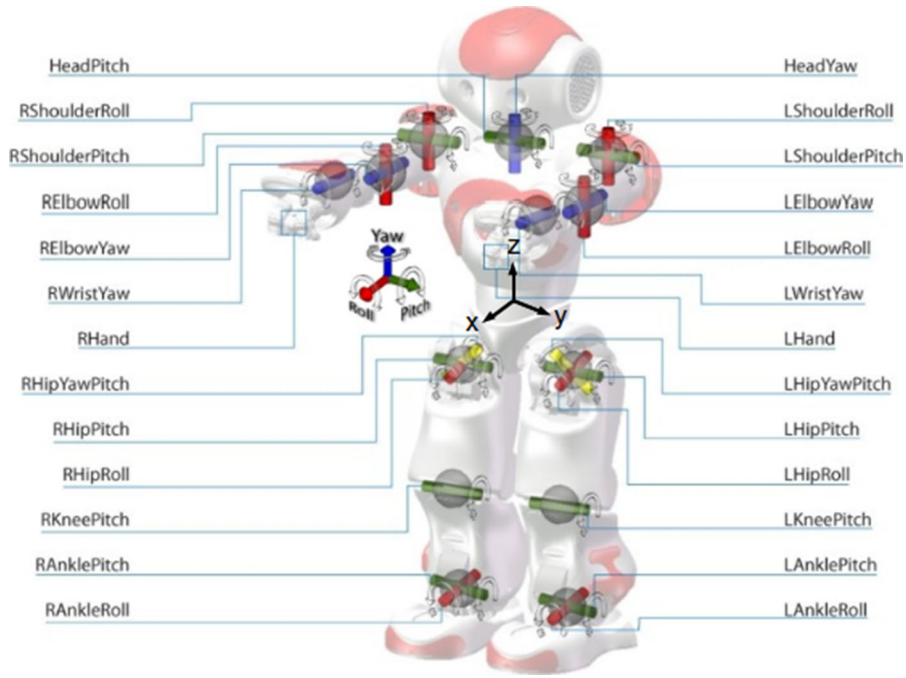

Fig. 1 Alebaran NAO H25 (v4.0) kinematic chains and joints (25 DOF) and base frame [5]

The 25 joints of NAO are organized into five kinematic chains as follows:

Head: HeadYaw, HeadPitch
Left Arm: LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll, LWristYaw, LHand
Right Arm: RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll, RWristYaw, RHand
Left Leg: LHipYawPitch, LHipRoll, LHipPitch, LKneePitch, LAnklePitch, LAnkleRoll
Right Leg: RHipYawPitch, RHipRoll, RHipPitch, RKneePitch, RAnklePitch, RAnkleRoll

LHipYawPitch and RHipYawPitch are just different names for the shared (common) joint (HipYawPitch) between the two legs. Figure 1 shows the physical arrangement of the five chains and their joints on the NAO robot. Note that the obsolete NAO H21 (RoboCup) edition is missing four DOF from the two hands (LWristYaw, LHand, RWristYaw, RHand). Clearly, the effective operation of the robot in the RoboCup soccer field requires precise control of all five manipulators. In particular, the head must be precisely controlled for effective camera operation, while the legs and arms must be controlled to generate walk motions and to perform manipulation of the environment (kicking, dribbling, etc.). Likewise, in any other robot application with the NAO, any kind of motion design for effective behavior will require precise control of one or more manipulators of the robot.

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

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

where $\mathbf{X} \in \Re^{3\times3}$ 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:

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{X}^{-1} & -\mathbf{X}^{-1}\bar{y} \\ [0 \cdots 0] & 1 \end{bmatrix} \tag{2}$$

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.

2.3 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 matrix 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 is always invertible.

2.4 Transformation Decomposition

The upper-left 3×3 block of an elementary transformation matrix (translation/rotation) is a rotation matrix by construction. The product $\mathbf{T}_3$ of two elementary transformations $\mathbf{T}_1$, $\mathbf{T}_2$ is:

$$\mathbf{T}_3 = \mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix}\mathbf{R}_1 & \bar{y}_1 \\ \bar{0}^\top & 1\end{bmatrix}\begin{bmatrix}\mathbf{R}_2 & \bar{y}_2 \\ \bar{0}^\top & 1\end{bmatrix} = \begin{bmatrix}\mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\bar{y}_2 + \bar{y}_1 \\ \bar{0}^\top & 1\end{bmatrix}$$

It is easy to validate that $\mathbf{R}_1\mathbf{R}_2$ is always a proper rotation. An arbitrary transformation matrix can be decomposed as a "translation after rotation" pair:

$$\mathbf{T} = \begin{bmatrix}\mathbf{R} & \bar{y} \\ \bar{0}^\top & 1\end{bmatrix} = \begin{bmatrix}\mathbf{I}_3 & \bar{y} \\ \bar{0}^\top & 1\end{bmatrix}\begin{bmatrix}\mathbf{R} & \bar{0} \\ \bar{0}^\top & 1\end{bmatrix} \tag{3}$$

At this point, we can extrapolate from Eqs. 2, 3 and reduce the computational complexity of inverting an arbitrary transformation matrix, utilizing the orthogonality of rotation matrices:

$$\begin{aligned}\mathbf{T}^{-1} = \begin{bmatrix}\mathbf{R} & \bar{0} \\ \bar{0}^\top & 1\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{I}_3 & \bar{y} \\ \bar{0}^\top & 1\end{bmatrix}^{-1} &= \begin{bmatrix}\mathbf{R}^\top & \bar{0} \\ \bar{0}^\top & 1\end{bmatrix}\begin{bmatrix}\mathbf{I}_3 & -\bar{y} \\ \bar{0}^\top & 1\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{R}^\top & -\mathbf{R}^\top\bar{y} \\ \bar{0}^\top & 1\end{bmatrix}\end{aligned} \tag{4}$$

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.

3 NAO Forward Kinematics Solution

Taking the torso frame of the NAO robot as the base frame (see Fig. 1), the forward kinematic equations for the five kinematic chains of NAO are the following:

$$\mathbf{T}_{\text{Base}}^{\text{Head}} = \mathbf{A}_{\text{Base}}^{0}\mathbf{T}_0^1\mathbf{T}_1^2\mathbf{R}_x(\tfrac{\pi}{2})\mathbf{R}_y(\tfrac{\pi}{2})\mathbf{A}_2^{\text{Head}} \tag{5}$$

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

$$\mathbf{T}_{\text{Base}}^{\text{RHand}} = \mathbf{A}_{\text{Base}}^{0}\mathbf{T}_0^1\mathbf{T}_1^2\mathbf{T}_2^3\mathbf{T}_3^4\mathbf{T}_4^5\mathbf{R}_x(\tfrac{\pi}{2})\mathbf{R}_z(\tfrac{\pi}{2})\mathbf{A}_5^{\text{RHand}} \tag{7}$$

$$\mathbf{T}_{\text{Base}}^{\text{LFoot}} = \mathbf{A}_{\text{Base}}^{0}\mathbf{T}_0^1\mathbf{T}_1^2\mathbf{T}_2^3\mathbf{T}_3^4\mathbf{T}_4^5\mathbf{T}_5^6\mathbf{R}_z(\pi)\mathbf{R}_y(-\tfrac{\pi}{2})\mathbf{A}_6^{\text{LFoot}} \tag{8}$$

$$\mathbf{T}_{\text{Base}}^{\text{RFoot}} = \mathbf{A}_{\text{Base}}^{0}\mathbf{T}_0^1\mathbf{T}_1^2\mathbf{T}_2^3\mathbf{T}_3^4\mathbf{T}_4^5\mathbf{T}_5^6\mathbf{R}_z(\pi)\mathbf{R}_y(-\tfrac{\pi}{2})\mathbf{A}_6^{\text{RFoot}} \tag{9}$$

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]. For the precise ordering and numbering of the joints in each chain, please refer to Section 2.1.

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 $\mathbf{T}_a^b = \left(\mathbf{T}_c^a\right)^{-1} \mathbf{T}_c^b$.

4 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 to 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 (they may fail, even if a solution exists) and poor performance [8]. On the other hand, analytical solutions are fast and exact, but extracting them takes significant effort.

4.1 Inverse Kinematics Methodology

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

4.1.1 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:

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

4.1.2 Construct the Symbolic Transformation

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

$$\mathbf{T}_{\text{base}}^0 \mathbf{T}_0^j(\theta_0, \ldots, \theta_j)\mathbf{T}_j^{\text{end}}$$

4.1.3 Form a Non-linear System

By equating the above transformation 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 the transformation matrix) of which only up to six are independent.

$$\mathbf{T} = \mathbf{T}_{\text{base}}^0 \mathbf{T}_0^j(\theta_0, \ldots, \theta_j)\mathbf{T}_j^{\text{end}}$$

4.1.4 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:

$$\left(\mathbf{T}_{\text{base}}^0\right)^{-1} \mathbf{T} \left(\mathbf{T}_j^{\text{end}}\right)^{-1} = \mathbf{T}_0^j(\theta_0, \ldots, \theta_j)$$

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

$$\mathbf{T}_c(\mathbf{T}_{\text{base}}^0)^{-1}\mathbf{T}(\mathbf{T}_j^{\text{end}})^{-1} = \mathbf{T}_c\mathbf{T}_0^j(\theta_0, \ldots, \theta_j)$$

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.

4.1.5 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 $\mathtt{a}$ and d parameters) or "arm and wrist" chains commonly found in humanoid robots, 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 2 shows the triangle formed by the upper 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.

4.1.6 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 ($asin$, $acos$). The transcendental nature of the $asin$ and $acos$ 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 the inverse trigonometric functions $atan$ and $acot$ are preferred, but forming them might not be possible for a particular chain.

4.1.7 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.

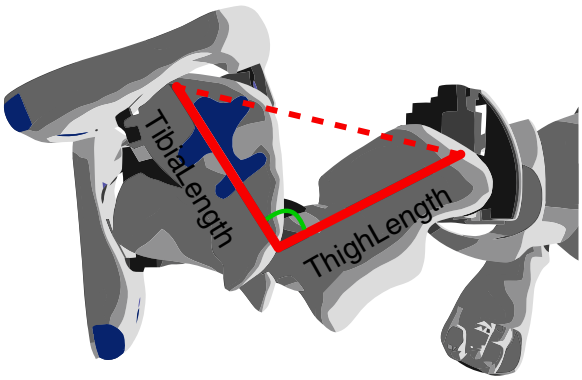

Fig. 2 Using geometry in the triangle formed by the robot leg to derive the knee joint angle

4.2 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.

5 NAO Inverse Kinematics Solution

Using the methodology presented in Section 4, we find the inverse kinematics solution for all five kinematic chains of NAO: head (2 joints), left arm (5 joints plus 1 toggle gripper), right arm, left leg (6 joints), and right leg. The solutions for the left chains are reused to construct solutions for the right chains by exploiting the symmetry in the sagittal body plane. Full derivation details may be found in a longer technical document [11]. Note that by setting the WristYaw and Hand joints on both arms to zero, the NAO H25 robot becomes identical to the restricted NAO H21 robot, whose kinematics were presented in our previous related work [12].

5.1 Inverse Kinematics for the Head Chain

The head chain consists of only two joints (HeadYaw (θ_1), HeadPitch (θ_2)), 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, assuming $a_x = 0$. In the former case, we construct the symbolic matrix through the forward kinematics solution (Eq. 5). 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

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

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

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

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 NAO Head inverse kinematics analytical solution

the end translation and l_3 is the z part of the base translation.

5.2 Inverse Kinematics for the Left Arm Chain

The left arm chain consists of five joints (LShoulderPitch (θ_1), LShoulderRoll (θ_2), LElbowYaw (θ_3), LElbowRoll (θ_4), LWristYaw (θ_5)). We ignore the last joint (LHand), since it is a simple toggle gripper and does not affect the kinematics. Before introducing our solution, we note that the left arm chain can be transformed to an equivalent chain by shifting the wrist joint along the lower arm limb without any change to the physical dynamics movement of the arm. Thus, we view the left arm chain as one in which the LWristYaw joint is placed immediately after the LElbowRoll joint (without any translation) and the translation between the LElbowRoll and LWristYaw joints is placed immediately after the LWristYaw joint.

The first three steps of our methodology (Sections 4.1.1, 4.1.2, 4.1.3) are straightforward given the forward kinematics solution (Eq. 6). Next, we remove the known base and end rotations and translations from both sides of the equation (Section 4.1.4) to make it simpler, so that we can identify solutions (Section 4.1.6):

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

By examining $\mathbf{T}'$, we see that θ_1 can be extracted from elements $\mathbf{T}'_{(1,4)}$ and $\mathbf{T}'_{(3,4)}$:

$$\left.\begin{array}{l}\mathbf{T}'_{(1,4)} = \cos\theta_1(l_2\cos\theta_2 + l_1\sin\theta_2) \\ \mathbf{T}'_{(3,4)} = -\sin\theta_1(l_2\cos\theta_2 + l_1\sin\theta_2)\end{array}\right\} \implies \theta_1 = atan\left(\frac{-\mathbf{T}'_{(3,4)}}{\mathbf{T}'_{(1,4)}}\right)$$

where l_1 = UpperArmLength and l_2 = ElbowOffsetY. Since the value of θ_1 is now known, we can remove the first joint from the chain:

$$\mathbf{T}'' = \left(\mathbf{T}^1_0\right)^{-1} \mathbf{T}'$$

Again, by a close examination of the translation part of the symbolic matrix, we see that we can use $\mathbf{T}''_{(1,4)}$ and $\mathbf{T}''_{(3,4)}$ to extract θ_2 (note that $l_2^2 + l_1^2 > 0$):

$$\left.\begin{array}{l}\mathbf{T}''_{(1,4)} = l_2\cos\theta_2 + l_1\sin\theta_2 \\ \mathbf{T}''_{(3,4)} = -l_1\cos\theta_2 + l_2\sin\theta_2\end{array}\right\} \implies$$

$$\theta_2 = \pm acos\left(\frac{l_2\mathbf{T}''_{(1,4)} - l_1\mathbf{T}'_{(3,4)}}{l_2^2 + l_1^2}\right)$$

Knowing θ_2, we manipulate the chain once more to remove the second joint:

$$\mathbf{T}''' = \left(\mathbf{T}^2_1\right)^{-1} \mathbf{T}''$$

We extract θ_3 from the symbolic matrix in a similar way (note that $\sin\theta_4 \neq 0$, since the LElbowYaw joint angle cannot reach 0 or $\pm\pi$, by construction):

$$\left.\begin{array}{l}\mathbf{T}'''_{(1,3)} = \cos\theta_3\sin\theta_4 \\ \mathbf{T}'''_{(3,3)} = \sin\theta_3\sin\theta_4\end{array}\right\} \implies \theta_3 = atan\left(\frac{\mathbf{T}'''_{(3,3)}}{\mathbf{T}'''_{(1,3)}}\right)$$

and then we remove the third joint from the chain:

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

Finally, we extract the values for the remaining two joints:

$$\left.\begin{array}{l}\mathbf{T}''''_{(1,3)} = \sin\theta_4 \\ \mathbf{T}''''_{(3,3)} = \cos\theta_4\end{array}\right\} \implies \theta_4 = atan\left(\frac{\mathbf{T}''''_{(1,3)}}{\mathbf{T}''''_{(3,3)}}\right)$$

$$\left.\begin{array}{l}\mathbf{T}''''_{(2,1)} = \sin\theta_5 \\ \mathbf{T}''''_{(2,2)} = \cos\theta_5\end{array}\right\} \Longrightarrow \theta_5 = atan\left(\frac{\mathbf{T}''''_{(2,1)}}{\mathbf{T}''''_{(2,2)}}\right)$$

The final step is to validate all candidate solutions through the forward kinematics validation step (Section 4.1.7) and discard all invalid sets of solutions. Figure 4 shows the resulting analytical solution, where $\mathbf{T}_{(i,j)}$ is the (i, j) element of matrix $\mathbf{T}$. An alternative approach for deriving the inverse kinematics of the arm, utilizing geometry, may be found in our previous work about the NAO H21 robot [12].

5.3 Inverse Kinematics for the Left Leg Chain

The kinematic chain of the left leg has six joints (LHipYawPitch (θ_1), LHipRoll (θ_2), LHipPitch (θ_3), LKneePitch (θ_4), LAnklePitch (θ_5), LAnkleRoll (θ_6)), but since the first three joints have intersecting axes, the problem is possibly solvable [9, 10]. We will show all the steps of our approach on this longest chain of the robot to illustrate the use of our generic methodology in finding inverse kinematics solutions.

First, we construct the numeric (Section 4.1.1) and the symbolic (Section 4.1.2) transformation matrices and then we form the non-linear system (Section 4.1.3). It is easy to see that we can simplify our system by manipulating of both sides (Section 4.1.4), namely by removing the known translations from the chain:

$$\widehat{\mathbf{T}} = \left(\mathbf{A}^0_{\text{Base}}\right)^{-1}\mathbf{T}\left(\mathbf{A}^{\text{End}}_6\right)^{-1}$$

Now, we have a chain from the base frame of the first joint to the (rotated) frame of the last joint. The first joint, LHipYawPitch (θ_1), is by construction rotated by $-\frac{3\pi}{4}$ about the x-axis with respect to the torso frame. Again we can manipulate both sides of the non-linear system (Section 4.1.4) and rotate the origin of the chain by $\frac{\pi}{4}$ about the x-axis to make the first joint a yaw joint (aligned with the z-axis):

$$\widetilde{\mathbf{T}} = \mathbf{R}_x(\tfrac{\pi}{4})\widehat{\mathbf{T}}$$

Now, we can observe that the first four joints (LHipYawPitch (θ_1), LHipRoll(θ_2), LHipPitch(θ_3), LKneePitch(θ_4)) affect the position and orientation of the end effector and the other two joints (LAnklePitch(θ_5), LAnkleRoll(θ_6)) affect only its orientation. It would be convenient, if only three joints were affecting the position of the end effector, since we operate in the three-dimensional space. Thus, we can manipulate again both sides (Section 4.1.4) and invert the transformation matrix to form the reverse chain. Now, only the LAnkleRoll(θ_6), LAnkle Pitch(θ_5), and LKneePitch(θ_4) joints affect the position:

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

The resulting symbolic matrix is still quite complex, but at this point we only need the translation block, which is relatively simple:

$$\begin{aligned}\mathbf{T}'_{(1,4)} &= l_2\sin\theta_5 - l_1\sin(\theta_4+\theta_5)\\ \mathbf{T}'_{(2,4)} &= \left(l_2\cos\theta_5 + l_1\cos(\theta_4+\theta_5)\right)\sin\theta_6\\ \mathbf{T}'_{(3,4)} &= \left(l_2\cos\theta_5 + l_1\cos(\theta_4+\theta_5)\right)\cos\theta_6\end{aligned}$$

where l_1 = ThighLength and l_2 = TibiaLength. We can now find θ_4 using the law of cosines (Section 4.1.5). We focus on the triangle formed by the leg with ThighLength, TibiaLength, and the dis-

Fig. 4 NAO Left Arm inverse kinematics analytical solution

$$\begin{array}{ll}\mathbf{T}' = \left(\mathbf{A}^0_{\text{Base}}\right)^{-1}\mathbf{T}\left(\mathbf{A}^{\text{End}}_4\right)^{-1}\left(\mathbf{R}_z(\tfrac{\pi}{2})\right)^{-1} & \\ \theta_1 = atan\left(\frac{-\mathbf{T}'_{(3,4)}}{\mathbf{T}'_{(1,4)}}\right) & \mathbf{T}'' = \left(\mathbf{T}^1_0\right)^{-1}\mathbf{T}' \\ \theta_2 = \pm acos\left(\frac{l_2\mathbf{T}''_{(1,4)} - l_1\mathbf{T}'_{(3,4)}}{l_2^2+l_1^2}\right) & \mathbf{T}''' = \left(\mathbf{T}^2_1\right)^{-1}\mathbf{T}'' \\ \theta_3 = atan\left(\frac{\mathbf{T}'''_{(3,3)}}{\mathbf{T}'''_{(1,3)}}\right) & \mathbf{T}'''' = \left(\mathbf{T}^3_2\right)^{-1}\mathbf{T}''' \\ \theta_4 = atan\left(\frac{\mathbf{T}''''_{(1,3)}}{\mathbf{T}''''_{(3,3)}}\right) & \theta_5 = atan\left(\frac{\mathbf{T}''''_{(2,1)}}{\mathbf{T}''''_{(2,2)}}\right)\end{array}$$

tance from the new base to the new end effector in the reverse chain as sides (see Fig. 2):

$$d = \sqrt{(s_x - p'_x)^2 + (s_y - p'_y)^2 + (s_z - p'_z)^2}$$

where $(s_x, s_y, s_z) = (0, 0, 0)$ is the new origin and $(p'_x, p'_y, p'_z) = \left(\mathbf{T}'_{(1,4)}, \mathbf{T}'_{(2,4)}, \mathbf{T}'_{(3,4)}\right)$ is the position of the new target point. Now, we use the law of cosines to find the interior angle θ'_4 between the thigh and tibia sides of the triangle:

$$\theta'_4 = acos\left(\frac{{l_1}^2 + {l_2}^2 - d^2}{2 l_1 l_2}\right)$$

Since θ'_4 represents an interior angle, whereas the LKneePitch joint is stretched in the zero position, the resulting angle θ''_4 in this range is computed by:

$$\theta''_4 = \pi - \theta'_4$$

Since the range of the LKneePitch joint includes both positive and negative angles, we finally extract θ_4 as:

$$\theta_4 = \pm\theta''_4$$

Next, we can solve a part of our system (Section 4.1.6) by extracting the θ_6 angle from the translation block using $\mathbf{T}'_{(2,4)}$ and $\mathbf{T}'_{(3,4)}$:

$$\frac{p'_y}{p'_z} = \frac{\mathbf{T}'_{(2,4)}}{\mathbf{T}'_{(3,4)}} \quad \Longrightarrow$$

$$\frac{p'_y}{p'_z} = \frac{\left(l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)\right)\sin\theta_6}{\left(l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_6)\right)\cos\theta_6} \quad \Longrightarrow$$

$$\theta_6 = atan\left(\frac{p'_y}{p'_z}\right), \text{ if } \left(l_2 \cos\theta_5 + l_1 \cos(\theta_4 + \theta_5)\right) \neq 0$$

Since both the nominator and the denominator of the result may become zero, the solution for θ_6 has some undefined points; these are discussed in Section 6.

To move on, we go back to $\widetilde{\mathbf{T}}$ and we remove (Section 4.1.4) the two rotations at the end of the chain along with the transformation $\mathbf{T}_5^6$ which is now known, because θ_6 is known:

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

As before, we form the reverse chain:

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

and we extract the new target position $(p''_x, p''_y, p''_z) = \left(\mathbf{T}''_{(1,4)}, \mathbf{T}''_{(2,4)}, \mathbf{T}''_{(3,4)}\right)$. The translation block of the new symbolic transformation matrix is:

$$\mathbf{T}''_{(1,4)} = l_2 \cos\theta_5 + l_1(\cos\theta_5 \cos\theta_4 - \sin\theta_5 \sin\theta_4)$$
$$\mathbf{T}''_{(2,4)} = -l_2 \sin\theta_5 - l_1(\sin\theta_5 \cos\theta_4 + \cos\theta_5 \sin\theta_4)$$
$$\mathbf{T}''_{(3,4)} = 0$$

In these equations, θ_5 is the only unknown. From $\mathbf{T}''_{(1,4)}$, we obtain an expression (Section 4.1.6) for $\cos\theta_5$:

$$\mathbf{T}''_{(1,4)} = p''_x \quad \Longrightarrow$$
$$(l_2 + l_1 \cos\theta_4)\cos\theta_5 = p''_x + l_1 \sin\theta_5 \sin\theta_4 \quad \Longrightarrow$$
$$\cos\theta_5 = \frac{p''_x + l_1 \sin\theta_5 \sin\theta_4}{l_2 + l_1 \cos\theta_4}$$
$$\text{if } (l_2 + l_1 \cos\theta_4) \neq 0$$

Given the lengths of the links of the robot, the denominator $l_2 + l_1 \cos\theta_4$ becomes zero, only if $\cos\theta_4 = -1.029$, which is impossible, since $|\cos\theta_4| \leq 1$. We continue with $\mathbf{T}''_{(2,4)}$:

$$\sin\theta_5(-l_2 - l_1 \cos\theta_4) - l_1 \cos\theta_5 \sin\theta_4 = p''_y \quad \Longrightarrow$$
$$\sin\theta_5(-l_2 - l_1 \cos\theta_4) - l_1 \frac{p''_x + l_1 \sin\theta_5 \sin\theta_4}{l_2 + l_1 \cos\theta_4}\sin\theta_4 = p''_y \quad \Longrightarrow$$
$$-\sin\theta_5(l_2 + l_1 \cos\theta_4) - \frac{l_1 p''_x \sin\theta_4}{l_2 + l_1 \cos\theta_4} - \frac{{l_1}^2 \sin\theta_5 \sin^2\theta_4}{l_2 + l_1 \cos\theta_4} = p''_y \quad \Longrightarrow$$
$$-\sin\theta_5(l_2 + l_1 \cos\theta_4)^2 - {l_1}^2 \sin\theta_5 \sin^2\theta_4 = p''_y(l_2 + l_1 \cos\theta_4) + l_1 p''_x \sin\theta_4 \quad \Longrightarrow$$
$$\theta_5 = asin\left(-\frac{p''_y(l_2 + l_1 \cos\theta_4) + l_1 p''_x \sin\theta_4}{{l_1}^2 \sin^2\theta_4 + (l_2 + l_1 \cos\theta_4)^2}\right)$$

The division is always feasible, because ${l_1}^2 \sin^2\theta_4 + (l_2 + l_1 \cos\theta_4)^2$ is obviously greater than zero for any value of θ_4. Now, we can go back to $\widetilde{\mathbf{T}}'$ and remove (Section 4.1.4) the two transformations $\mathbf{T}_3^4$ and $\mathbf{T}_4^5$, since θ_4 and θ_5 are known:

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

The translation block in the transformation $\mathbf{T}'''$ must be zero, because the only joints left are the three hip

joints, which only affect the orientation. The rotation block of the transformation is:

$$\mathbf{T}'''_{(1,1)} = \cos\widehat{\theta}_1 \cos\widehat{\theta}_2 \cos\theta_4 - \sin\widehat{\theta}_1 \sin\theta_3$$
$$\mathbf{T}'''_{(1,2)} = -\cos\theta_3 \sin\widehat{\theta}_1 - \cos\widehat{\theta}_1 \cos\widehat{\theta}_2 \sin\theta_3$$
$$\mathbf{T}'''_{(1,3)} = \cos\widehat{\theta}_1 \sin\widehat{\theta}_2$$
$$\mathbf{T}'''_{(2,1)} = -\cos\theta_3 \sin\widehat{\theta}_2$$
$$\mathbf{T}'''_{(2,2)} = \sin\widehat{\theta}_2 \sin\theta_3$$
$$\mathbf{T}'''_{(2,3)} = \cos\widehat{\theta}_2$$
$$\mathbf{T}'''_{(3,1)} = -\cos\widehat{\theta}_2 \cos\theta_3 \sin\widehat{\theta}_1 - \cos\widehat{\theta}_1 \sin\theta_3$$
$$\mathbf{T}'''_{(3,2)} = -\cos\widehat{\theta}_1 \cos\theta_3 + \cos\widehat{\theta}_2 \sin\widehat{\theta}_1 \sin\theta_3$$
$$\mathbf{T}'''_{(3,3)} = -\sin\widehat{\theta}_1 \sin\widehat{\theta}_2$$

where $\widehat{\theta}_1$ is the DH parameter θ for the first (LHipYawPitch) joint and $\widehat{\theta}_2$ is the DH parameter θ for the second (LHipRoll) joint. Now, we can extract the remaining three angles as follows (Section 4.1.6):

$$\widehat{\theta}_2 = acos\left(\mathbf{T}'''_{(2,3)}\right)$$
$$\theta_2 = \widehat{\theta}_2 - \frac{\pi}{4}$$
$$\theta_3 = asin\left(\frac{\mathbf{T}'''_{(2,2)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right)$$
$$\widehat{\theta}_1 = acos\left(\frac{\mathbf{T}'''_{(1,3)}}{\sin\left(\theta_2 + \frac{\pi}{4}\right)}\right)$$
$$\theta_1 = \widehat{\theta}_1 + \frac{\pi}{2}$$

The LHipRoll joint (θ_2) cannot physically reach $-\frac{\pi}{4}$ or $\frac{3\pi}{4}$. Therefore the denominator $\sin\left(\theta_2 + \frac{\pi}{4}\right)$ can never be zero. At this point, we have obtained all candidate solutions (vectors of joint values). The wrong ones are discarded by applying the validation step (Section 4.1.7) and, thus, only valid solutions are returned (it is possible to end up with more than one valid solutions). The inverse kinematics equations for the left leg are summarized in Fig. 5.

5.4 Inverse Kinematics for the Right Side

A keen observation for a humanoid robot is that there typically exists a symmetry with respect to the sagittal body plane. Clearly, if a particular target position is reachable by a left-side manipulator, there exists a symmetrical target position reachable by the corresponding right-side manipulator through a mirrored configuration. Let -$\mathbf{M} = diag([1\ -1\ 1\ 1])$ be the reflection that maps points in the three-dimensional space to the symmetric ones with respect to the sagittal plane (normal to the y-axis). Note that $\mathbf{M}^{-1} = \mathbf{M}$. Given a target transformation $\mathbf{T}_\mathrm{R}$ we seek to reach with a right-side limb, it's easy to see that $\mathbf{T}_\mathrm{L} = \mathbf{M}\mathbf{T}_\mathrm{R}\mathbf{M}$ is a proper rigid body transformation that can be reached by the corresponding left-side limb. Therefore, the analytical solutions for the left-side chains can be reused for the right-side chains. To mirror back the resulting joint configuration(s), we exploit the symmetries in the physical construction of NAO. For the arms, the joint values map naturally to opposite values for the ShoulderRoll, ElbowYaw, ElbowRoll, and WristYaw joints and identical value for the ShoulderPitch joint. For the legs, the symmetry emerges for opposite values of the HipRoll and AnkleRoll joints with the values of the remaining joints being identical. More formally:

$$\mathbf{T}^{\mathrm{LHand}}_{\mathrm{Base}}(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5) = \mathbf{M}\,\mathbf{T}^{\mathrm{RHand}}_{\mathrm{Base}}(\theta_1, -\theta_2, -\theta_3, -\theta_4, -\theta_5)\,\mathbf{M}$$
$$\mathbf{T}^{\mathrm{LLeg}}_{\mathrm{Base}}(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = \mathbf{M}\,\mathbf{T}^{\mathrm{RLeg}}_{\mathrm{Base}}(\theta_1, -\theta_2, \theta_3, \theta_4, \theta_5, -\theta_6)\,\mathbf{M}$$

6 Implementation

Having completed all kinematics in analytical form, we created `NAOKinematics`, a software library for real-time, on-board 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` [13] 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 for the inverse kinematics of each chain; their input is the desired target position and the output is a set of solutions, where each one contains values for all the joints of the specified chain. Although multiple solutions are rare, all valid solutions found are returned, so that the user can decide which one to use. Finally, the library 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 target positions for the legs which lead to an undefined AnkleRoll joint,

Fig. 5 NAO Left Leg inverse kinematics analytical solution

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

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

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

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

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

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

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

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

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

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

when the KneePitch and AnklePitch joints take very 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 in the relevant configuration subspace is shown in Fig. 7. To check if these configurations ever occur in practice, 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 the resulting motion trajectories alongside the problematic locus in Fig. 7. It is clear that no motion brought the robot to these configurations. In fact, unlimited numerical precision is

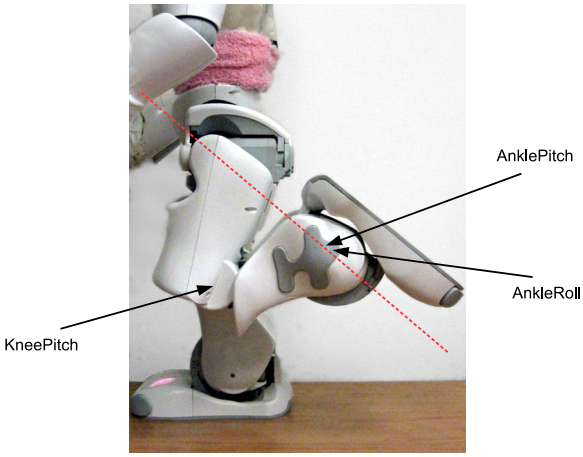

Fig. 6 An instance of the problematic leg configurations

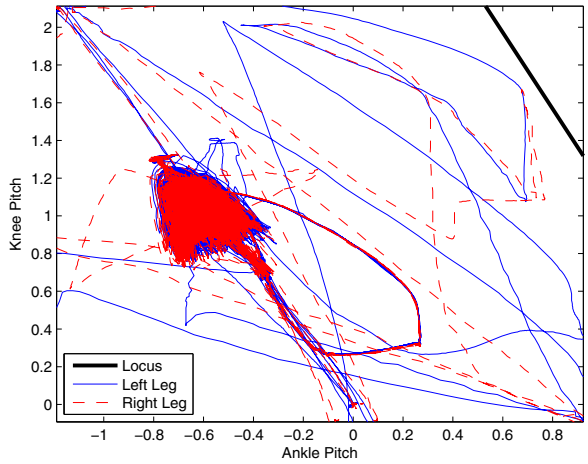

Fig. 7 Trajectories of motion in a subspace of the leg joints

required to "hit" the locus, because even a small perturbation within the machine precision can lead to a regular solution. Due to finite precision computations, it is practically impossible to force the required value to be exactly zero. In addition, it is rather unlikely that anyone will consistently give target positions that drive the joints in that locus. Nevertheless, in our implementation we chose to raise an error flag, so that the case is detected, if it occurs, and handled at a higher level. A simple solution for such cases is to skip the current actuation loop and leave the actuators at their current setting. Assuming a smooth planned trajectory of target positions and a typical high-frequency actuation loop (50 to 100Hz on the NAO), the problematic instance will go unnoticed and the motion will proceed smoothly.

Due to the physical coupling of the two HipYawPitch joints, it is possible that the desired target transformations for the legs are not both feasible, even when separate solutions are found. The coupling imposes that the yaw orientations of the legs are symmetrical. When this holds, if solutions exist for both legs, they will be mutually consistent; the two HipYawPitch values will be identical. It is a good practice to manually enforce this constraint at a higher level by providing target positions accordingly. In any case, the user is called to choose which solution takes precedence. For example, a robot balancing on the left leg may benefit from choosing the HipYawPitch values from the support (left) leg solution, even if that causes the right leg to deviate from the desired trajectory.

To compare and contrast our analytical solution with a commonly-used numerical approach, a Newton gradient-descent iterative solver for inverse kinematics was implemented. We formulate the Jacobian pseudo-inverse problem [8] using the analytical forward kinematics equations (Section 3). For a given target transformation matrix, 12 equations are to be solved (a rotation matrix and a translation vector) with respect to the free variables corresponding to joint values. An iterative solution approach is to form a linearized system for these equations around a given point in the joint space and use a gradient-descent method to move to a point closer to a solution. Using the DH transformation for an N-DOF chain, we compute the N partial derivatives of the transformation with respect to all variables to form the Jacobian matrix of the chain. The locally linearized $12 \times N$ system is solved using the Moore-Penrose pseudo-inverse. This Newton gradient-descent algorithm guarantees quadratic rate of convergence, when the current point is near a solution. Unfortunately, when the solution of the problem is not unique, saddle points in the joint space exist, in which the algorithm fails, being unable to select a descent direction. Furthermore, the iteration may converge to a solution that violates the physical constraints of the chain. The final solution depends on the initialization and while this ensures that the nearest solution in the joint space is reached, it also implies that a known (feasible) solution might not be reached in favor of another (possibly infeasible) one. This iterative solver can be used, complementary to the analytical solution, to solve for target transformations that are not strictly feasible or in parallel for other kinematic chains.

7 Results

7.1 Real-Time Performance

One of our goals was to implement an optimized software library for real-time kinematics computations on-board the robot. We measured the performance of our analytical method for each of the functions we offer, as well as the performance of the Jacobian method for low ($\epsilon < 10^{-2}$) and high ($\epsilon < 10^{-6}$) accuracy. Table 1 shows average on-board execution times in microseconds (μs). Clearly, the analytical method offers significant speed-ups compared to the iterative one with high or low accuracy.

7.2 Demonstration I: Basic Center-of-Mass Balancing

In this demonstration, we seek to implement a basic balancing method using our analytical kinematics. We make NAO move 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, angleY) of the torso plane and we calculate the translation of the CoM relatively to the rotated torso:

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

Table 1 On-board mean execution times in μs for various kinematics calls on NAO H25 (v4.0)

Kinematics Function	Robot Chain	Analytical (Exact)	Jacobian (High Accuracy)	Jacobian (Low Accuracy)
Forward	Head	9.65	–	–
Forward	Arm Left	16.80	–	–
Forward	Arm Right	16.90	–	–
Forward	Leg Left	18.00	–	–
Forward	Leg Right	18.00	–	–
Inverse	Head	19.20	21.35	15.25
Inverse	Arm Left	50.01	498.21	302.80
Inverse	Arm Right	51.31	497.82	306.14
Inverse	Leg Left	53.62	557.94	328.29
Inverse	Leg Right	54.01	559.37	325.41
CoM	All	50.90	–	–

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

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

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. On some robots the foot is not entirely parallel to the floor, due to a displaced inertial unit (accelerometers and gyro-meters), not due to kinematics.

7.3 Demonstration II: Pointing to a Moving Ball

In this demonstration, our goal is to make NAO point to a moving ball with its stretched arms. Apart from our analytical kinematics, to realize this task we employed our vision module [13] 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 has been found, the robot raises only one arm; the other arm cannot physically point to the ball.

7.4 Demonstration III: Human-Guided Balancing on Two Legs

In this demonstration, a human operator interacts with the robot using its left shoulder as a kind of joystick. The joint values of the arm are used to compute a desired posture. The user can command the robot to raise or lower its torso and to move its center of mass forward or backward. Our goal is to drive the legs using inverse kinematics. This simple demo is used to physically assess and compare different inverse kinematics methods on the two legs. In our setup, the right leg uses the numerical Jacobian method to find a solution around the current posture, while the left leg uses the analytical solution, which requires no initialization. When the user guides the robot into a region of

the joint space where the Jacobian method fails systematically, the right leg fails to respond to user input as the iterative solver gets trapped, unlike the left leg which responds normally.

8 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 initialization dependencies and singularities due to the proposed methodology. In addition, we contributed an implementation of the proposed NAO kinematics as a freely-available software library for real-time execution on the real or simulated robots.

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 v4.0 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 failures and, therefore, lacks in robustness. Similar results were obtained with own numerical approach. It should be noted that parts of the three demonstrations could not 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 omnidirectional walk algorithms, dynamic balancing methods, dynamic kick engines, etc. Our library offers 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. 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.

Acknowledgments We would like to thank all (past and present) members of RoboCup team Kouretes for their help.

References

1. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., Matsubara, H.: Robocup: a challenge problem for AI. AI Mag. **18**(1), 73–85 (1997)
2. Gouaillier, D., Blazevic, P.: A mechatronic platform, the Aldebaran Robotics humanoid robot. In: Proceedings of the 32nd IEEE Annual Conference on Industrial Electronics (IECON), pp. 4049–4053 (2006)
3. Graf, C., Härtl, A., Röfer, T., Laue, T.: A robust closed-loop gait for the Standard Platform League humanoid. In: Proceedings of the 4th Workshop on Humanoid Soccer Robots, pp. 30–37 (2009)
4. Jadidi, M.G., Hashemi, E., Harandi, M.A.Z., Sadjadian, H.: 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. Only available online: www.aldebaran-robotics.com/documentation (2012)
6. Denavit, J., Hartenberg, R.S.: A kinematic notation for lower-pair mechanisms based on matrices. ASME J. Appl. Mech. **22**, 215–221 (1955)
7. Hartenberg, R.S., Denavit, J.: Kinematic Synthesis of Linkages. McGraw-Hill, New York (1964)
8. Buss, S.R.: Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least-squares methods. Available at: www.math.ucsd.edu/∼sbuss/ResearchWeb/ikmethods/iksurvey.pdf (2009)
9. Pieper, D., Roth, B.: The kinematics of manipulators under computer control. In: Proceedings of the 2nd International Congress on Theory of Machines and Mechanisms, vol. 2, pp. 159–169 (1969)
10. Pieper, D.: The kinematics of manipulators under computer control. PhD. thesis, Stanford University (1968)
11. Kofinas, N.: Forward and inverse kinematics for the NAO humanoid robot. Diploma thesis, Technical University of Crete, Greece. Available at: www.intelligence.tuc.gr/lib/downloadfile.php?id=430 (2012)
12. Kofinas, N., Orfanoudakis, E., Lagoudakis, M.G.: Complete analytical inverse kinematics for NAO. In: Proceedings of the 13th International Conference on Autonomous Robot Systems and Competitions (ROBOTICA) (2013)
13. Orfanoudakis, E.: Reliable object recognition for the RoboCup domain. Diploma thesis. Technical University of Crete, Greece (2011)