Real Time Inverse Kinematics Robot Gait Sequencing
Using Self Organizing Maps

Chris Jordan G. Aliac

Intelligent Systems Laboratory, Cebu Institute of Technology, Cebu City, 6000 Philippines
chrisjord26@yahoo.com

Abstract. Real-time control of end-effector of humanoid robots in Cartesian coordinates requires computationally efficient solutions, especially when the complexity of analytic solutions becomes too high along with the number of degrees-of-freedom. In this study, we solve the inverse kinematics problem for dynamic gait sequencing combining mathematical modeling and neural networks and applying this to an actual Biped Robot. Forward kinematics transforms robotic configurations in joint space, into end-effector locations in Cartesian space. Pseudo-learning is done by associating the Cartesian space into the joint space until convergence. Our learning strategy is based on the topology-preserving property of the Extended Kohonen Model both in the competitive and output layers, even in rather high-dimensional spaces. The virtual output angles of the self-organizing maps are then mapped to the corresponding servomotor actuation values before being transmitted to the robot for control. The theoretical and actual results are illustrated with an integrated 3D simulation system and a 10 degree-of-freedom biped robot employing Bluetooth communication technology and Microsoft SAPI speech software in task issuance, respectively.

1 Introduction

The analytic approach in inverse-kinematics is first to derive the forward-kinematics equations \( p = f(\theta) \), where \( p \) is a vector of Cartesian coordinates of the robot's end-effector, and \( \theta \) is a vector of joint variables, and then solve for the inverse-kinematics equations \( \theta = f^{-1}(p) \).

Finding such linear equations, if \( f(\theta) \) is not uniquely invertible at all, requires much mathematical experience and precise knowledge of the robot's mechanical structure as well. No general solutions exist, thus, the term pseudo-inverse functions apply to the inverse-kinematics equations. In addition, the computational complexity greatly increases with the robot's degree-of-freedom, as is the case with the 10 degree-of-freedom biped robot used in this study.
Prerequisite to the process of learning inverse kinematics is a framework for deriving training data. Since in this study the purpose of solving inverse kinematics is dynamic gait sequencing, gaits are defined based on kinematic trajectories in Cartesian space, and it is therefore vital to have a mapping $\theta \rightarrow p$. Before transformations from joint to Cartesian spaces are realized, a mathematical model of the robot must be laid out. This is with the usage of DH Parameters $d_i$, $a_i$, $\alpha_i$, and $\theta_i$, describing relationships between links of a kinematic chain, where $\theta_i$ are variables and are initialized into “home position” values.

Table 1. -1 : DH Parameters for the robot in study

<table>
<thead>
<tr>
<th></th>
<th>Left leg (L)</th>
<th></th>
<th></th>
<th>Right leg (R)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$d_i$</td>
<td>$a_i$</td>
<td>$\alpha_i$</td>
<td>$\theta_i$</td>
<td>$d_i$</td>
<td>$a_i$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>63</td>
<td>$90^\circ$</td>
<td>$180^\circ$</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>45</td>
<td>$270^\circ$</td>
<td>$270^\circ$</td>
<td>10</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>50</td>
<td>$0^\circ$</td>
<td>$21^\circ$</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>63</td>
<td>$0^\circ$</td>
<td>$339^\circ$</td>
<td>0</td>
<td>63</td>
</tr>
<tr>
<td>5</td>
<td>-8</td>
<td>46</td>
<td>$90^\circ$</td>
<td>$0^\circ$</td>
<td>-8</td>
<td>47</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>25</td>
<td>$0^\circ$</td>
<td>$0^\circ$</td>
<td>10</td>
<td>25</td>
</tr>
</tbody>
</table>
Fig. 2. Mathematical Model

With DH parameters, the homogenous frame transforms \( i-1P_i \) for every link frame \( F_i \) in the kinematic chain takes the form:

\[
\begin{bmatrix}
    c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\
    s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\
    0 & s\alpha_i & c\alpha_i & d_i \\
    0 & 0 & 0 & 1
\end{bmatrix}
\]

The forward kinematic equations for the end-effector locations are finally derived as a product of successive homogenous frame transforms in the link:

\[
0_{P_n} = 0_{P_1} 1_{P_2} ... n-1_{P_n}
\]
where

\[ ^6p_x = \begin{bmatrix} n_x & s_x & t_x & p_x \\ n_y & s_y & t_y & p_y \\ n_z & s_z & t_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

In the case of our biped robot, the values for the elements of the homogenous frame transforms are as follows:

\[
\begin{align*}
n_{Lx} &= -c_{\theta L2} c(\theta L3 + \theta L4 + \theta L5) c_{\theta L6} + s_{\theta L2} s_{\theta L6} \\
n_{Ly} &= -s(\theta L3 + \theta L4 + \theta L5) c_{\theta L6} \\
n_{Lz} &= s_{\theta L2} c(\theta L3 + \theta L4 + \theta L5) c_{\theta L6} + c_{\theta L2} s_{\theta L6} \\
n_{Rx} &= c_{\theta R2} c(\theta R3 + \theta R4 + \theta R5) c_{\theta R6} + s_{\theta R2} s_{\theta R6} \\
n_{Rx} &= s(\theta R3 + \theta R4 + \theta R5) c_{\theta R6} + c_{\theta R2} s_{\theta R6} \\
n_{Rz} &= -c_{\theta R2} c(\theta R3 + \theta R4 + \theta R5) s_{\theta R6} - c_{\theta R2} c_{\theta R6} \\
n_{Rx} &= -c_{\theta R2} s(\theta R3 + \theta R4 + \theta R5) \\
n_{Rz} &= s_{\theta R2} s(\theta R3 + \theta R4 + \theta R5) \\
\end{align*}
\]
\[ p\text{Rx} = \left[ aR6 c(\thetaR3 + \thetaR4 + \thetaR5) c\thetaR6 + aR5 c(\thetaR3 + \thetaR4 + \thetaR5) + aR4 c(\thetaR3 + \thetaR4) + aR3 c\thetaR3 + aR2 - dR6 s(\thetaR3 + \thetaR4 + \thetaR5) \right] c\thetaR2 + aR1 + (dR5 + dR4 + dR3 - aR6 s\thetaR6) s\thetaR2 \]

\[ p\text{Ry} = aR6 s(\thetaR3 + \thetaR4 + \thetaR5) c\thetaR6 + aR5 s(\thetaR3 + \thetaR4 + \thetaR5) + aR4 s(\thetaR3 + \thetaR4) + aR3 s\thetaR3 + dR6 c(\thetaR3 + \thetaR4 + \thetaR5) + dR2 \]

\[ p\text{Rz} = -\left[ aR6 c(\thetaR3 + \thetaR4 + \thetaR5) c\thetaR6 + aR5 c(\thetaR3 + \thetaR4 + \thetaR5) + aR4 c(\thetaR3 + \thetaR4) + aR3 c\thetaR3 + aR2 - dR6 s(\thetaR3 + \thetaR4 + \thetaR5) \right] s\thetaR2 + (dR5 + dR4 + dR3 - aR6 s\thetaR6) c\thetaR2 \]

In our approach, only the position vector \( p \) is relevant to inverse kinematics pseudo-learning, wherein \( p = f([\theta2 \theta3 \theta4 \theta5 \theta6]) \) is a function of 5 joint variables for each leg, disregarding \( \theta1 \), which is a constant. As training data, the Cartesian coordinates \( p = (px, py, pz) \) are the input signals and the joint values \( \theta = [\theta2 \theta3 \theta4 \theta5 \theta6] \) are the associated target output values.

Ranges for the actuation values of servomotors are exactly \([1, 254]\), while ranges for the angular values in the mathematical model vary. Offline control of servomotors require that virtual joint values be mapped into its corresponding actuator value, since the subject of learning are the virtual joint values. The following generic one-to-one mapping are proposed, wherein four parameters, namely home virtual angle \( \theta_v \), home, home actual angle \( \theta_a \), home, maximum actual angle \( \theta_{a, max} \), and minimum actual angle \( \theta_{a, min} \), are defined. There are two types of mapping, the parallel and the anti-parallel mapping, wherein the former, both values in the actual and virtual angle ranges increase as the actuation goes to the same direction, while in the latter, both values in the actual and virtual angle ranges increase as the actuation goes to the opposite direction. Derivation is based on discrete mathematics.

2 Inverse Kinematics Pseudo-learning

The Extended Kohonen Model consists of an input layer, competitive layer, and an output layer, and its mode of learning is supervised learning due to the introduction of target output values together with input signals during training. In this study, two self-organizing maps were used, one for each leg, of which the rationale is to be able to segregate the learning process for each leg, since the intersection of the task spaces of each leg is not the null space, the implication of which is an event we term as “confusion”. For illustration, a Cartesian point reached by the left leg can also be reached by the right leg, but the joint values are different. Upon introduction of these inputs, the single self-organizing map will memorize output values that are conflicting, and the result during deployment will be an ill-posed kinematic chain.

The dimensions of the task space vary according to the DH parameters. Thus the number of neural units in the three dimensional neural networks must be dynamic and computed depending on the set of training data \( G = \{(px, py, pz) 02 03 04 05 06\} \) since the set training data is composed of Cartesian coordinates and corresponding joint angles in the task space. A constant \( r \) is also defined, which is the number of neurons assigned to subspaces of size \( r^3 \) distance units.
task space is represented by two Cartesian coordinates $p_{\text{min}} = (px,_{\text{min}}, py,_{\text{min}}, pz,_{\text{min}})$ and $p_{\text{max}} = (px,_{\text{max}}, py,_{\text{max}}, pz,_{\text{max}})$.

From $p_{\text{min}}$ and $p_{\text{max}}$, the dimensions $b$, $d$, and $h$ of the neural networks are computed.

$$b = \text{ceil}( \frac{|\text{floor}(x_{\text{min}} - a)| + \text{floor}(x_{\text{max}} + a)|}{r})$$

$$d = \text{ceil}( \frac{|\text{floor}(y_{\text{min}} - a)| + \text{floor}(y_{\text{max}} + a)|}{r})$$

$$h = \text{ceil}( \frac{|\text{floor}(z_{\text{min}} - a)| + \text{floor}(z_{\text{max}} + a)|}{r})$$

where $|n|$ computes the absolute value of $n$, ceil and floor are the usual discrete integer functions.

Parameters to the training phase include initial learning rate $\eta_0$, initial
neighborhood size $\delta_0$, input bias $x_{inp}$, output bias $x_{out}$, and maximum number of epochs $T$. The learning algorithms are as enlisted below:

1. Initialize node connection weights with random values in the range $(0, 1)$.
2. For all training data $g = [px \ yz \ 02 \ 03 \ 04 \ 05 \ 06 \ s]$, where $s$ indicates which leg is in scrutiny, locate the neuron $h_{i,j,k}$ responsible for the input signal $p = [px \ yz]$ and initialize output weights with $\theta = [\theta_2 \ \theta_3 \ \theta_4 \ \theta_5 \ \theta_6]$ where,

$$i = \text{floor}(\frac{px - \text{floor}(px, \min)}{r})$$
$$j = \text{floor}(\frac{py - \text{floor}(py, \min)}{r})$$
$$k = \text{floor}(\frac{pz - \text{floor}(pz, \min)}{r})$$

3. For each epoch $t: 0 \rightarrow T$
   1. For $p: 0 \rightarrow n$
      1. Find winning neuron $r' = \text{argmin}_{i,j,k} || p - w ||^2$
      2. For all neuron $r$, update $w_r = t^{-1}w_r + \eta_d r' (p - t^{-1}w_r)$
      3. For all neuron $r$, update $\theta'_r = t^{-1}\theta'_r + \eta_d r' (\theta - t^{-1}\theta'_r)$
   2. Update learning rate $\eta_t = (1 - t/T)\eta_{t-1}$
   3. Update neighborhood size $\delta_t = (1 - t/T)\delta_{t-1}$, where

$$drr' = e^{-(|| p - w ||^2 / 2\delta_t^2)}$$

and $n$ is the number of training data $g$.

Average input and output weights error per epoch $t$ are inspected during training, since graphs asymptotic to zero could possibly indicate convergence on competitive and/or output layers. Computations are as follows.

$$\varepsilon_{\text{input},k} = \frac{n\sum_{i=1}^{n} || p_i - w_i ||^2}{n}$$
$$\varepsilon_{\text{output},k} = \frac{n\sum_{i=1}^{n} || \theta_i - \theta'_i ||^2}{n}$$

3 **Simulation Systems**

Three inter-related simulation systems were designed for this study namely, the Gaits Designer, the Humanoid Robot Training Simulator, and the Intelligent Humanoid Robot Behaviour, implemented in the object-oriented language C#.
In the training phase, gait phases are created, and the joint values are input into the Gaits Designer. These gait phases can be stored as a .gts file for later manipulation or as a .trd file for training. The Humanoid Robot Training Simulator then loads the .trd file and starts training after learning parameters have been set. The resulting weights can then finally be stored as a .kts file and loaded in the Intelligent Humanoid Robot Behaviour for deployment. Voice commands issued are recognized by the SAPI voice tool, and corresponding dynamic gait sequences are invoked. After mapping virtual to actual values, actuation values for each servomotor are transmitted to the biped robot.

All three simulation systems render three dimensional viewing of the kinematic chain, self-organizing maps, task space samples, and other entities, using a transform matrix $P$ for display of 3D Cartesian coordinates.

$$P = \begin{bmatrix}
    \cos \gamma \cos \beta & \gamma \sin \beta \cos \alpha - \gamma \cos \alpha & \gamma \sin \beta \sin \alpha + \gamma \cos \alpha & 0 \\
    \sin \gamma \cos \beta & \gamma \sin \beta \cos \alpha + \gamma \cos \alpha & \gamma \sin \beta \sin \alpha - \gamma \cos \alpha & 0 \\
    -\sin \beta & \sin \beta \cos \alpha & \sin \beta \sin \alpha & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}$$

Fig. 5. Figure 3-1: Training Phase Subsystems Architecture
Moreover, to display 3D coordinates in 2D graphics, an angle $\phi$, aside from display parameters $\alpha$, $\beta$, and $\gamma$, is employed, where $x' = x + y \cos \phi$ and $y' = z + y \sin \phi$, such that $(x, y, z)$ is a 3D coordinate and $(x', y')$ is a 2D coordinate.

References

4. Chung-Hsien Kuo, Yao-Chun Hung, En-Chou Lin, Jian-Chih Chen, Mike Lee, Chuan-Chia Sun. Autonomous Humanoid Soccer Robot: Taiwan 101 VI. Graduate Institute of Medical Mechatronics, Chang Gung University, Taiwan. Aaron D'Souza, Sethu Vijaya- kumar and Stefan Schaal. Learning Inverse Kinematics. Computer Science and Neuroscience, HNB-103, Univ. of Southern California, Los Angeles, CA 90089-2520, Kawato
5. Dynamic Brain Project (ERATO/JST), 2-2 Hikaridai, Seika-cho, Soraku-gun, 619-02 Kyoto, Japan.
7. Sears and Zemansky's University Physics. Young, Hugh D. 2005
11. http://davis.wpi.edu/~matt/courses/soms/