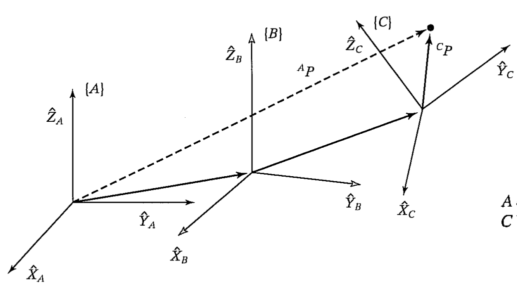
Inverse Kinematics of HOAP-2 robot

by

Dr. Rawichote Chalodhorn (Choppy)

Humanoid Robotics Lab, Neural System Group, Dept. of Computer Science & Engineering, University of Washington.

Coordinate Transformations



$$^{A}P = {}^{A}_{B}T\,{}^{B}_{C}T\,{}^{C}P$$

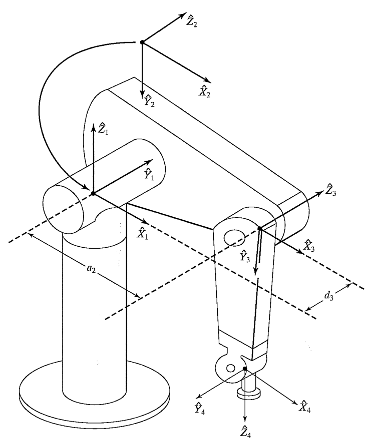
$$^{A}_{C}T = {}^{A}_{B}T\,{}^{B}_{C}T$$

$$^{A}_{C}T = \left[\begin{array}{ccc|c} \multicolumn{3}{c|}{{}^{A}_{B}R\,{}^{B}_{C}R} & {}^{A}_{B}R\,{}^{B}P_{CORG} + {}^{A}P_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$$^{B}_{A}T = {}^{A}_{B}T^{-1}$$

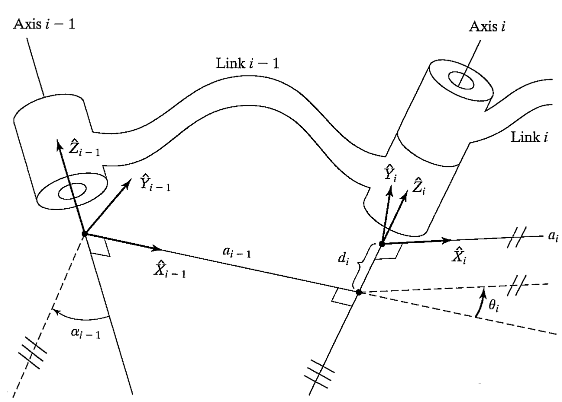
$$^{B}_{A}T = \left[\begin{array}{ccc|c} \multicolumn{3}{c|}{{}^{A}_{B}R^{T}} & -{}^{A}_{B}R^{T}\,{}^{A}P_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

PUMA 560 Kinematics Structure



i	$\alpha_i - 1$	$a_i - 1$	d_i	θi
1	0	0	0	θ_1
2	−90°	0	0	θ_2
3	0	a_2	d_3	θ_3
4	−90°	a_3	d_4	θ_4
5	90°	0	0	θ_5
6	−90°	0	0	θ_6

The Denavit and Hartenberg (D-H) convention



$a_i =$ *the distance from* $\hat{Z}_i$ *to* $\hat{Z}_{i+1}$ *measured along* $\hat{X}_i$;

$\alpha_i =$ *the angle from* $\hat{Z}_i$ *to* $\hat{Z}_{i+1}$ *measured about* $\hat{X}_i$;

$d_i =$ *the distance from* $\hat{X}_{i-1}$ *to* $\hat{X}_i$ *measured along* $\hat{Z}_i$; and

$\theta_i =$ *the angle from* $\hat{X}_{i-1}$ *to* $\hat{X}_i$ *measured about* $\hat{Z}_i$.

The Transformation Matrices

$$
{}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_2 & -c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{2}_{3}T = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & a_2 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{3}_{4}T = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & a_3 \\ 0 & 0 & 1 & d_4 \\ -s\theta_4 & -c\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{4}_{5}T = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_5 & c\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{5}_{6}T = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s\theta_6 & -c\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
$$

$$
{}^{4}_{6}T = {}^{4}_{5}T\,{}^{5}_{6}T = \begin{bmatrix} c_5c_6 & -c_5s_6 & -s_5 & 0 \\ s_6 & c_6 & 0 & 0 \\ s_5c_6 & -s_5s_6 & c_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{3}_{6}T = {}^{3}_{4}T\,{}^{4}_{6}T = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & -c_4s_5 & a_3 \\ s_5c_6 & -s_5s_6 & c_5 & d_4 \\ -s_4c_5c_6 - c_4s_6 & s_4c_5s_6 - c_4c_6 & s_4s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
$$

$$
{}^{1}_{3}T = {}^{1}_{2}T\,{}^{2}_{3}T = \begin{bmatrix} c_{23} & -s_{23} & 0 & a_2c_2 \\ 0 & 0 & 1 & d_3 \\ -s_{23} & -c_{23} & 0 & -a_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{1}_{6}T = {}^{1}_{3}T\,{}^{3}_{6}T = \begin{bmatrix} {}^{1}r_{11} & {}^{1}r_{12} & {}^{1}r_{13} & {}^{1}p_x \\ {}^{1}r_{21} & {}^{1}r_{22} & {}^{1}r_{23} & {}^{1}p_y \\ {}^{1}r_{31} & {}^{1}r_{32} & {}^{1}r_{33} & {}^{1}p_z \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

The Direct Kinematics

$$
{}^{0}_{6}T = {}^{0}_{1}T\,{}^{1}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

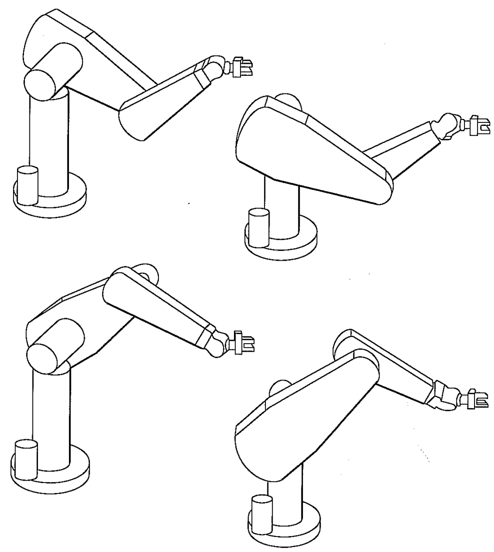
$$
\begin{aligned}
r_{11} &= c_1[c_{23}(c_4c_5c_6 - s_4s_5) - s_{23}s_5c_5] + s_1(s_4c_5c_6 + c_4s_6), \\
r_{21} &= s_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6 - c_1(s_4c_5c_6 + c_4s_6), \\
r_{31} &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6,
\end{aligned}
$$

$$
\begin{aligned}
r_{12} &= c_1[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] + s_1(c_4c_6 - s_4c_5s_6), \\
r_{22} &= s_1[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] - c_1(c_4c_6 - s_4c_5s_6), \\
r_{32} &= -s_{23}(-c_4c_5s_6 - s_4c_6) + c_{23}s_5s_6,
\end{aligned}
$$

$$
\begin{aligned}
r_{13} &= -c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5, \\
r_{23} &= -s_1(c_{23}c_4s_5 + s_{23}c_5) + c_1s_4s_5, \\
r_{33} &= s_{23}c_4s_5 - c_{23}c_5,
\end{aligned}
$$

$$
\begin{aligned}
p_x &= c_1[a_2c_2 + a_3c_{23} - d_4s_{23}] - d_3s_1, \\
p_y &= s_1[a_2c_2 + a_3c_{23} - d_4s_{23}] + d_3c_1, \\
p_z &= -a_3s_{23} - a_2s_2 - d_4c_{23}.
\end{aligned}
$$

Non-unique Solutions of Inverse Kinematics



The Jacobians

$$y_1 = f_1(x_1, x_2, x_3, x_4, x_5, x_6),$$
$$y_2 = f_2(x_1, x_2, x_3, x_4, x_5, x_6),$$
$$\vdots$$
$$y_6 = f_6(x_1, x_2, x_3, x_4, x_5, x_6).$$

$$\delta y_1 = \frac{\partial f_1}{\partial x_1}\delta x_1 + \frac{\partial f_1}{\partial x_2}\delta x_2 + \cdots + \frac{\partial f_1}{\partial x_6}\delta x_6,$$
$$\delta y_2 = \frac{\partial f_2}{\partial x_1}\delta x_1 + \frac{\partial f_2}{\partial x_2}\delta x_2 + \cdots + \frac{\partial f_2}{\partial x_6}\delta x_6,$$
$$\vdots$$
$$\delta y_6 = \frac{\partial f_6}{\partial x_1}\delta x_1 + \frac{\partial f_6}{\partial x_2}\delta x_2 + \cdots + \frac{\partial f_6}{\partial x_6}\delta x_6,$$

$$Y = F(X).$$

$$\delta Y = \frac{\partial F}{\partial X}\delta X.$$

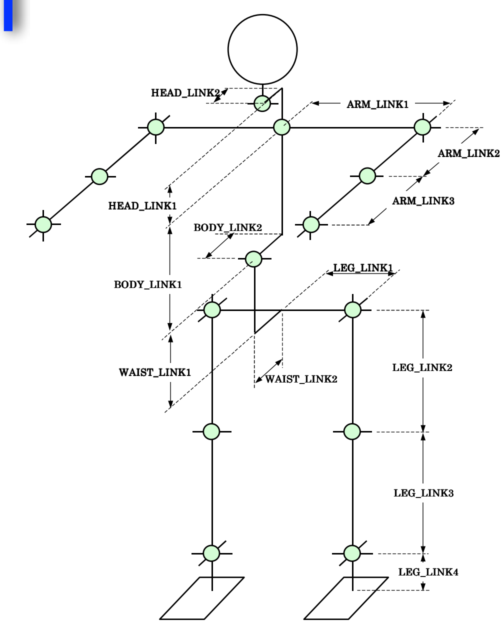
$$\delta Y = J(X)\delta X.$$

$$\dot{Y} = J(X)\dot{X}.$$

$$^0\nu = {}^0J(\Theta)\dot{\Theta},$$

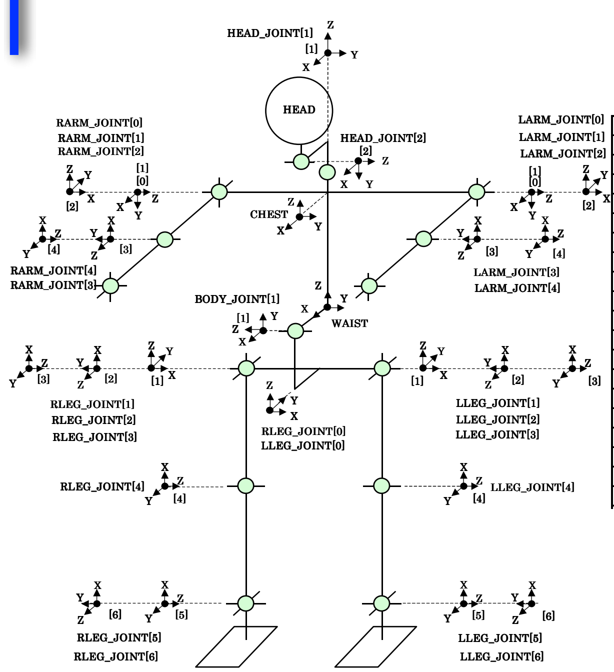
$$\dot{\Theta} = J^{-1}(\Theta)\nu.$$

Outline



ARM_LINK1	0.0995	m
ARM_LINK2	0.1010	m
ARM_LINK3	0.1460	m
LEG_LINK1	0.0390	m
LEG_LINK2	0.1000	m
LEG_LINK3	0.1000	m
LEG_LINK4	0.0370	m
BODY_LINK1	0.0900	m
BODY_LINK2	0.0340	m
HEAD_LINK1	0.0810	m
HEAD_LINK2	0.0080	m
WAIST_LINK1	0.0550	m
WAIST_LINK2	0.0340	m

Outline



	a[I−1]	α[I−1]	d[i]	θ [i]
	(m)	(deg)	(m)	(deg)
CHEST	−BODY_LINK2	0	BODY_LINK1	0
HEAD_JOINT[1]	0	0	HEAD_LINK1	0
HEAD_JOINT[2]	HEAD_LINK2	−90	0	0
ARM_JOINT[0]	0	−90	(*)ARM_LINK1	0
ARM_JOINT[1]	0	0	0	θ [1]
ARM_JOINT[2]	0	90	0	θ [2]+90
ARM_JOINT[3]	0	90	ARM_LINK2	θ [3]+90
ARM_JOINT[4]	0	90	0	θ [4]
BODY_JOINT[1]	0	90	0	0
LEG_JOINT[0]	−WAIST_LINK2	−90	−WAIST_LINK1	90
LEG_JOINT[1]	(*)LEG_LINK1	0	0	θ [1]
LEG_JOINT[2]	0	90	0	θ [2]+90
LEG_JOINT[3]	0	90	0	θ [3]
LEG_JOINT[4]	−LEG_LINK2	0	0	θ [4]
LEG_JOINT[5]	−LEG_LINK3	0	0	θ [5]
LEG_JOINT[6]	0	−90	0	θ [6]

Sample MATLAB code for calculate inverse kinematic of left arm of HOAP-2 robot

```
function bodyJoints = inverseKinectmatic(data)


%Note that these codes looks dumb, because I just want to get it done!

%Load Denavit-Hartenberg Parameters of the robot
load HOAP2KinematicModel

pose = getJointStructXYZ(data);

%Cal IK for left arm

%Cal tranformation matrix of the neck with respect to the world
 T_armRef = getArmRefFrame(data);


%Cal tranformation matrix of arm joint0
armJoint0.d = norm(pose.arm.L.shoulder - pose.neck);
T_AJ0 = getTransformaitonMatrix(armJoint0);

jointsXYZ = [pose.arm.L.elbow pose.arm.L.hand];

%Calculate local joint postions with resepect to shoulder frame
jointsXYZLocal = invT(T_armRef * T_AJ0) * [jointsXYZ; ones(1,2)];
jointsXYZLocal(4,:) = [];
r = jointsXYZLocal(:,1);

%Calculate joint1 and joint2 (shoulder joints) by using spherical
coordinate system
%conversion formulae
joint1 = rad2deg(atan2(r(2),r(1)));
joint2 = rad2deg(acos(r(3)/norm(r)));
joint2 = 180 - joint2;

%Cal2culate the elbow joint by using the cosine's law
C = norm(jointsXYZLocal(:,2));
l2 = norm(jointsXYZLocal(:,2)-jointsXYZLocal(:,1));
l1 = norm(jointsXYZLocal(:,1));
joint4 = -rad2deg(acos((C^2 - l1^2 - l2^2)/(2*l2*l1)));

% This method of calculating joint3 is commented out because it creates
% discontinuity in the solution
% %Calculate joint3 from analytic IK solution
% joint3 = acosd(-(- l1*cosd(joint2) - l2*cosd(joint2)*cosd(joint4) -
jointsXYZLocal(3,2))/(- l2*sind(joint2)*sind(joint4)));
% %Check value of joint3, 'coz the cosine function in MATLAB only
defined
% %from 0 to pi. Joint3 can be -pi to - 2*pi.
```

```
% hand_y = l1*sind(joint1)*sind(joint2) + l2*sind(joint4)*(cosd(joint1)
*sind(joint3)...
%     - sind(joint1)*cosd(joint2)*cosd(joint3)) + l2*sind(joint1)*cosd
(joint4)*sind(joint2);
% if (abs(hand_y-jointsXYZLocal(2,2))>0.0001)
%     joint3 = -joint3;
% end

%Update the transformation matrices of joint1 and joint2
armJoint1.theta = joint1;
armJoint2.theta = joint2;
T_AJ1 = getTransformaitonMatrix(armJoint1);
T_AJ2 = getTransformaitonMatrix(armJoint2);

%Calculate the hand postion with resepect to the transformation matrix
of joint 4
hand_T3 = invT(T_armRef * T_AJ0 * T_AJ1 * T_AJ2 * T_AJ3) * [jointsXYZ(:,
2); 1];

%This part is for handling numerical error
if (abs(hand_T3(2))<0.0001)
    hand_T3(2) = 0;
end
if (abs(hand_T3(1))<0.0001)
    hand_T3(1) = 0;
end

%Calculate joint3 by using a simple atan formular since the forarm is
now
%appeared to be a vector that is pointing out of the origin of frame 3
joint3 = rad2deg(atan2(hand_T3(2),hand_T3(1))) + 90;

bodyJoints.arm.L.joint1 = joint1;
bodyJoints.arm.L.joint2 = joint2;
bodyJoints.arm.L.joint3 = joint3;
bodyJoints.arm.L.joint4 = joint4;
bodyJoints.arm.L.joint5 = 0;
bodyJoints.arm.L.link1 = armJoint0.d;
bodyJoints.arm.L.link2 = l1;
bodyJoints.arm.L.link3 = l2;
```