Inside vnoid

written by Y Tazaki

Preface

- Inverse kinematics calculation shown in the following pages is derived by myself. I did not consult any textbook.
- There are many other ways (including numerical methods) to solve the same problem. Check out yourself.
- I don't guarantee that there are no flaws in my derivation. Use it under your own responsibility. Verify the derivation by yourself.

- IK demos are included in vnoidlib.
  Run them on Choreonoid to see how they actually work.

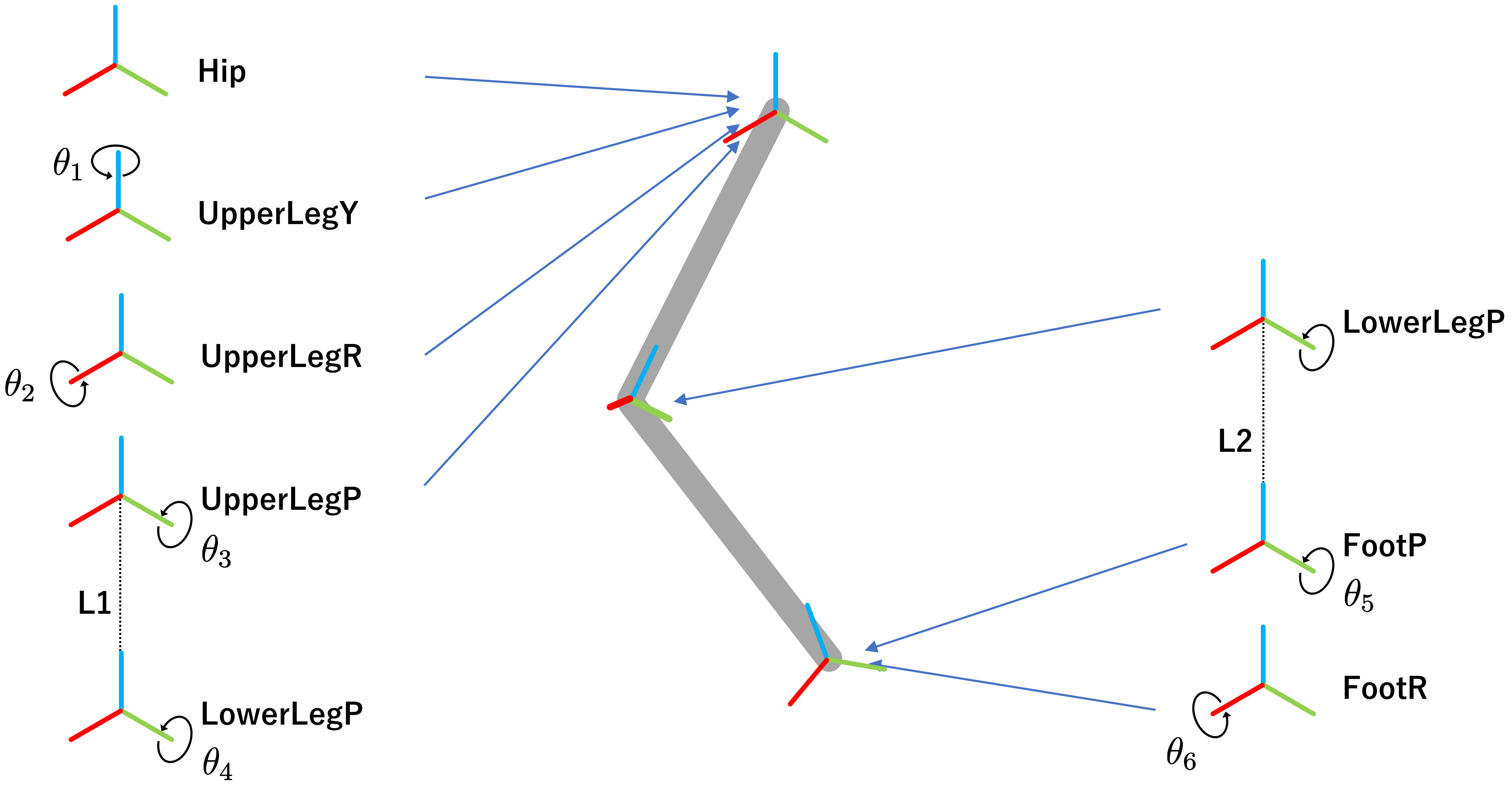
Kinematics

- Leg IK
  - An analytical solution of a YRPPPR-type kinematic chain is derived.
  - Assumption:
    - No offset between three hip joints
    - No offset between two ankle joints

  - Another solution for the same kinematic chain is shown in Kajita’s book:
    ヒューマノイドロボット 改訂2版

Kinematics

We consider the following YRPPPR –type kinematic chain.



Kinematics

Using affine transformation, the forward kinematics from the **Hip** to the **Foot** is expressed as follows.

$$R_{\mathrm{z}}(\theta_1)R_{\mathrm{x}}(\theta_2)R_{\mathrm{y}}(\theta_3)T_{\mathrm{z}}(-L_1)R_{\mathrm{y}}(\theta_4)T_{\mathrm{z}}(-L_2)R_{\mathrm{y}}(\theta_5)R_{\mathrm{x}}(\theta_6)$$

$$R_{\mathrm{x}}(\theta) = \begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \qquad T_{\mathrm{x}}(l) = \begin{bmatrix} I & \begin{bmatrix} l \\ 0 \\ 0 \end{bmatrix} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

$$R_{\mathrm{y}}(\theta) = \begin{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \qquad T_{\mathrm{y}}(l) = \begin{bmatrix} I & \begin{bmatrix} 0 \\ l \\ 0 \end{bmatrix} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

$$R_{\mathrm{z}}(\theta) = \begin{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \qquad T_{\mathrm{z}}(l) = \begin{bmatrix} I & \begin{bmatrix} 0 \\ 0 \\ l \end{bmatrix} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

Kinematics

Inverse Kinematics *See CompLegIk of vnoidlib for actual implementation.*

Consider that the relative position and rotation of the foot with respect to the hip are given, and we would like to calculate the joint angles.

Relative position of the foot is:

$$\boldsymbol{p} = \begin{bmatrix} p_\mathrm{x} \\ p_\mathrm{y} \\ p_\mathrm{z} \end{bmatrix}$$

Relative rotation of the foot is expressed by Euler angles.

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_\mathrm{x} \\ \theta_\mathrm{y} \\ \theta_\mathrm{z} \end{bmatrix}$$

The equivalent rotation matrix is:

$$R = R_\mathrm{z}(\theta_\mathrm{z})R_\mathrm{y}(\theta_\mathrm{y})R_\mathrm{x}(\theta_\mathrm{x})$$

Kinematics

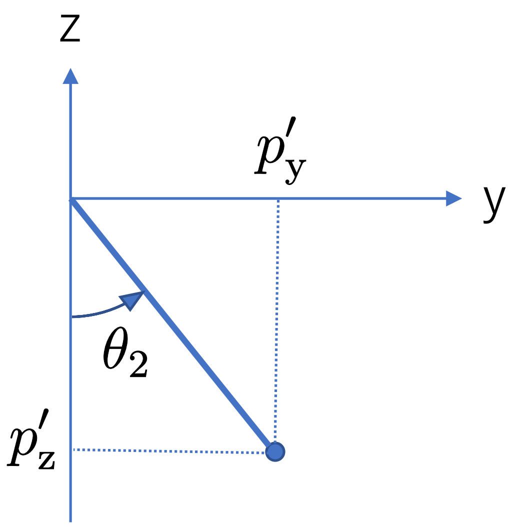
The hip-yaw angle is immediately determined by the yaw angle of the foot.

$$\theta_1 = \theta_\mathrm{z}$$

Now, let us express the foot position in the local coordinate frame of **UpperLegY**.

$$\boldsymbol{p}' = R_\mathrm{z}(\theta_1)^\mathsf{T}\boldsymbol{p}$$

Consider the projection on the y-z plane of this local coordinate frame.



Then, the hip-roll angle is obtained by

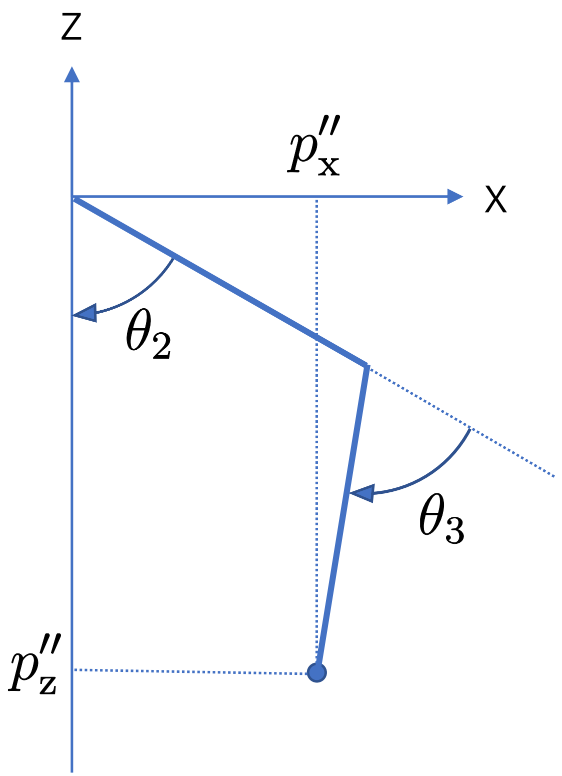
$$\theta_2 = \mathrm{atan2}(p'_\mathrm{y}, -p'_\mathrm{z})$$

Kinematics

Next, let us express the foot position in the local coordinate frame of **UpperLegR**.

$$\boldsymbol{p}'' = R_{\mathrm{x}}(\theta_2)^{\mathsf{T}}\boldsymbol{p}'$$

Here, the hip, the knee, and the ankle all lie on the x-z plane of this local coordinate frame.



Kinematics

Define angles as shown in the right figure.

Using trigonometry, we get

$$\alpha = -\mathrm{atan2}(p''_{\mathrm{x}}, -p''_{\mathrm{z}})$$

$$\beta = \mathrm{acos}\left(\frac{L_1^2 + L_2^2 - d^2}{2L_1L_2}\right) \qquad d = \sqrt{{p''}_{\mathrm{x}}^2 + {p''}_{\mathrm{z}}^2}$$

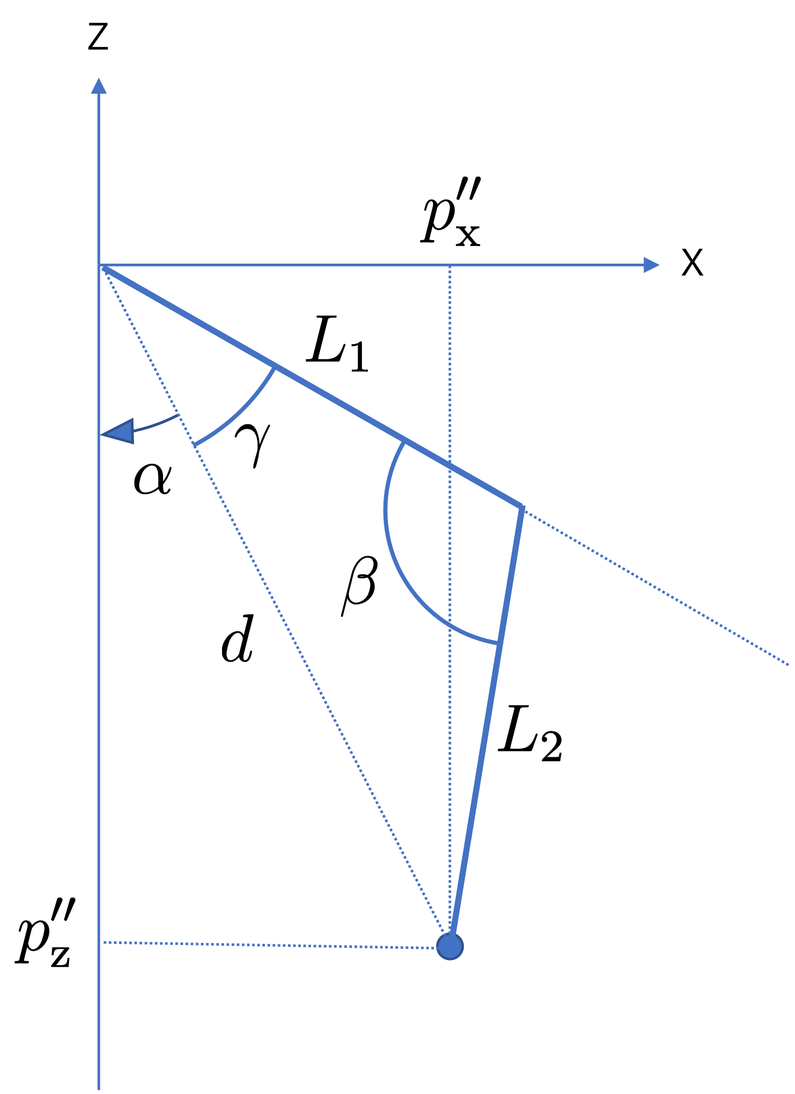
$$\gamma = \mathrm{asin}\left(\frac{L_2\sin(\beta)}{d}\right)$$

and thus

$$\theta_2 = \alpha - \gamma$$

$$\theta_3 = \pi - \beta$$

Note that we can easily detect singular postures (knee gets stretched) by monitoring the argument of **acos**.
See the actual implementation for details.



Kinematics

Now, let us determine the ankle-pitch and ankle-roll angles.

The relative rotation from **LowerLegP** to **FootR** is given by

$$R' = (R_{\mathrm{z}}(\theta_1)R_{\mathrm{x}}(\theta_2)R_{\mathrm{y}}(\theta_3)R_{\mathrm{y}}(\theta_4))^{\mathsf{T}} R$$

Calculate Euler angles equivalent to this rotation.

$$\boldsymbol{\theta}' = \mathsf{rot2rpy}(R')$$

See ToRollPitchYaw of vnoidlib for implementation of this function.

Here, the yaw rotation angle is always 0.

Using the pitch and roll angles, we get:

$$\theta_5 = \theta'_{\mathrm{y}}$$

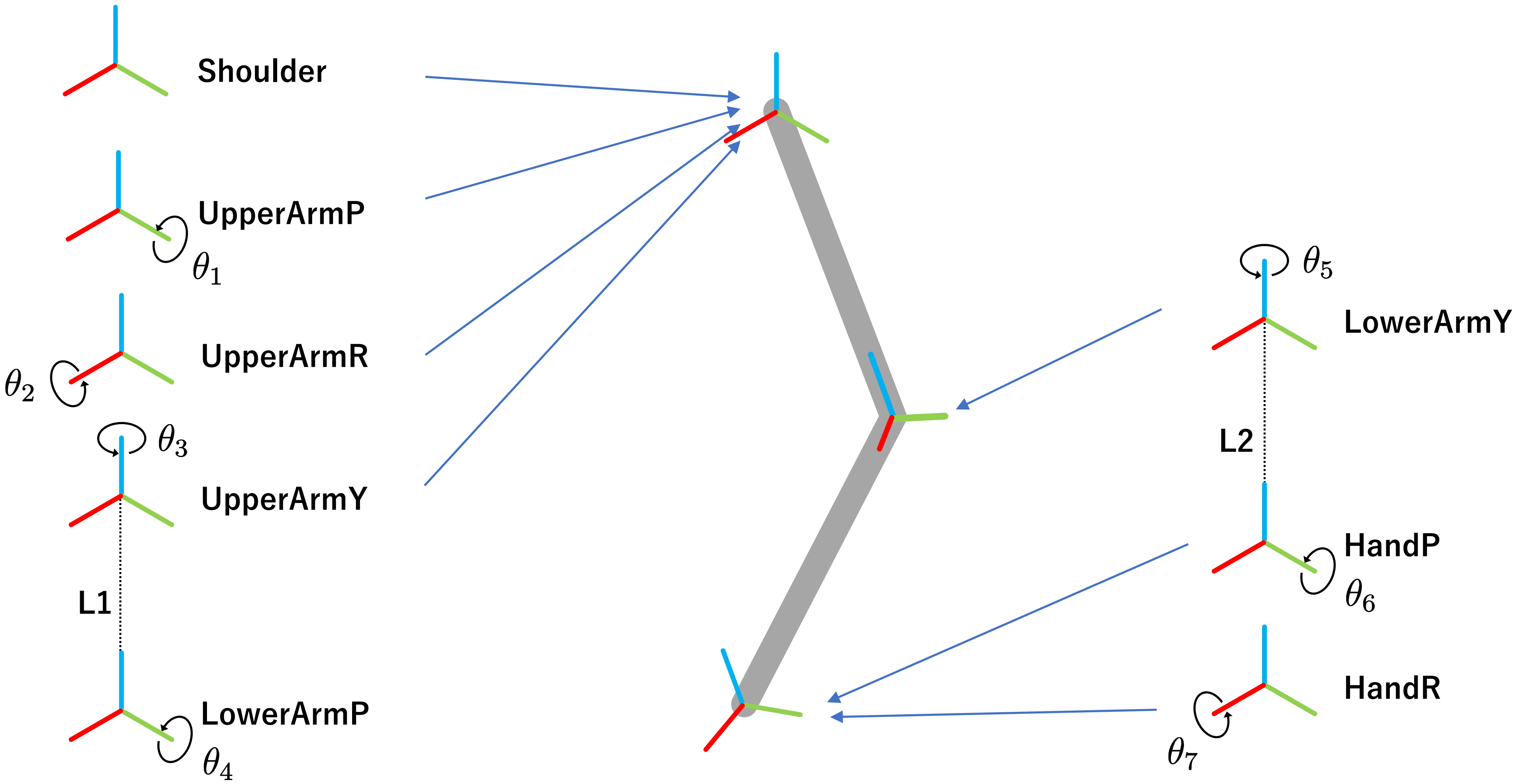
$$\theta_6 = \theta'_{\mathrm{x}}$$

Kinematics

- Arm IK

Kinematics

We consider the following PRYPYPR –type kinematic chain.



Kinematics

Solving Arm-IK is harder than Leg-IK, since there are 7 joints and kinematics is redundant.

One way to make it simple is to require the user to directly specify the angle of one joint, and solve IK for remaining 6 joints.

In the following, we consider that the shoulder-yaw angle is specified by the user.

Relative position of the hand is:

$$\boldsymbol{p} = \begin{bmatrix} p_{\mathrm{x}} \\ p_{\mathrm{y}} \\ p_{\mathrm{z}} \end{bmatrix}$$

Relative rotation of the hand is expressed by Euler angles.

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_{\mathrm{x}} \\ \theta_{\mathrm{y}} \\ \theta_{\mathrm{z}} \end{bmatrix}$$

The equivalent rotation matrix is:

$$R = R_{\mathrm{z}}(\theta_{\mathrm{z}}) R_{\mathrm{y}}(\theta_{\mathrm{y}}) R_{\mathrm{x}}(\theta_{\mathrm{x}})$$

Kinematics

As mentioned previously, the shoulder-yaw angle is given.

$$\theta_3 \quad \text{given}$$

The bend angle of the elbow is obtained by trigonometry.

$$\beta = \operatorname{acos}\left(\frac{L_1^2 + L_2^2 - d^2}{2L_1L_2}\right) \qquad d = \|\boldsymbol{p}\|$$

$$\theta_4 = -(\pi - \beta)$$

Note the negative sign here. This is because the elbow bends in the negative direction in our setup.

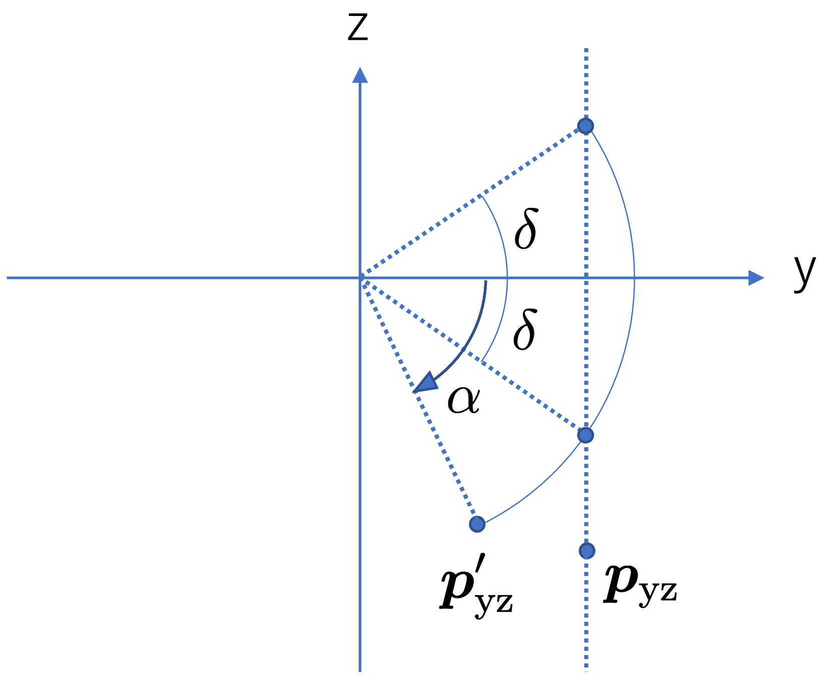
Similarly to the Leg-IK, singular configurations (elbow gets stretched) can be detected by monitoring the argument of **acos**.

Kinematics

Consider a hand position when the shoulder pitch and roll angles are both zero.

$$\boldsymbol{p}' = R_{\mathrm{z}}(\theta_3)\left(\begin{bmatrix}0\\0\\-L_1\end{bmatrix} + R_{\mathrm{y}}(\theta_4)\begin{bmatrix}0\\0\\-L_2\end{bmatrix}\right)$$

Now, project this point and the desired hand position on the y-z plane.



$$\alpha = \mathrm{atan2}(p'_{\mathrm{z}}, p'_{\mathrm{y}})$$

$$\delta = \mathrm{acos}\left(\frac{p_{\mathrm{y}}}{\sqrt{p_{\mathrm{y}}^2 + p_{\mathrm{z}}^2}}\right)$$

Kinematics

We would like to rotate $\boldsymbol{p}'_{\mathrm{yz}}$ around the x-axis so that, after rotation, its y-coordinate matches that of $\boldsymbol{p}_{\mathrm{yz}}$.

Therefore we get:

$$\theta_2 = -\alpha \pm \delta$$

Here, you must choose one from two possible solutions. One way is to choose depending on the sign of the z coordinate of the desired hand position.

$$\theta_2 = \begin{cases} -\alpha + \delta & \text{if } p_{\mathrm{z}} > 0 \\ -\alpha - \delta & \text{otherwise} \end{cases}$$

Kinematics

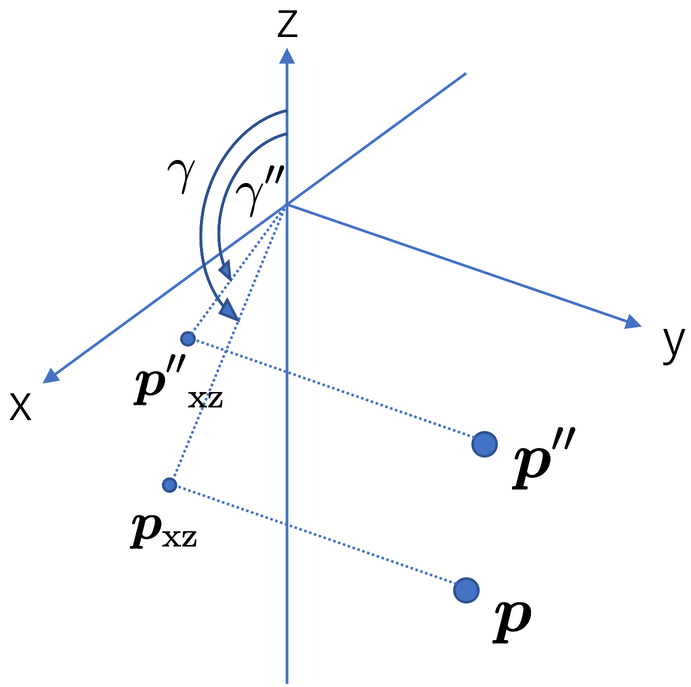
Next, consider a hand position when only the shoulder pitch angle is zero.

$$\boldsymbol{p}'' = R_{\mathrm{x}}(\theta_2)\boldsymbol{p}'$$

Now, project this point and the desired hand position on the x-z plane, and define angles as shown in the figure.

$$\gamma = \mathrm{atan2}(p_{\mathrm{x}}, p_{\mathrm{z}})$$

$$\gamma'' = \mathrm{atan2}({p''}_{\mathrm{x}}, {p''}_{\mathrm{z}})$$



The shoulder pitch angle is the difference of these angles.

$$\theta_1 = \gamma - \gamma''$$

Kinematics

Now, let us determine the wrist yaw-pitch-roll angles.

The relative rotation from **LowerArmP** to **HandR** is given by

$$R' = (R_{\mathrm{y}}(\theta_1)R_{\mathrm{x}}(\theta_2)R_{\mathrm{z}}(\theta_3)R_{\mathrm{y}}(\theta_4))^{\mathsf{T}}R$$

Calculate Euler angles equivalent to this rotation.

$$\boldsymbol{\theta}' = \mathsf{rot2rpy}(R')$$

See ToRollPitchYaw of vnoidlib for implementation of this function.

Using these angles, we get:

$$\theta_5 = \theta'_{\mathrm{z}}$$

$$\theta_6 = \theta'_{\mathrm{y}}$$

$$\theta_7 = \theta'_{\mathrm{x}}$$

Trajectory Generation

Ground Reaction Force Control

Low-level Control