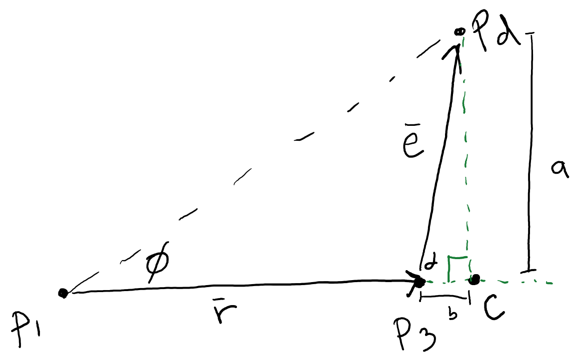
IK (Part 2)

Thursday, November 11, 2021 12:47 PM

Last class, we discussed a method based on euler angles for computing the orientation of p1 (ex. "shoulder").

<u>Alternate method</u> (based on tangent)



<u>Goal</u>: point the direction r towards Pd

Let $\bar{e} = P_d - P_3$

$\bar{r} = p_3 - p_1$

ϕ = angle between $\bar{r}$ + $(P_d - P_1)$

<u>Insight</u>: Form a right triangle $P_1 C P_d$ and right triangle $P_3 C P_d$

Let α be an angle in $P_3 C P_d$

Let a, b be lengths

$$\sin\alpha = \frac{a}{\|e\|}, \quad \cos\alpha = \frac{b}{\|e\|} \Rightarrow \begin{array}{l} a = \|e\| \sin\alpha \\ b = \|e\| \cos\alpha \end{array}$$

$$\tan\phi = \frac{a}{b + \|r\|} = \frac{\|e\| \sin\alpha}{\|e\| \cos\alpha + \|r\|} \quad \text{multiply by } \frac{\|r\|}{\|r\|}$$

$$= \frac{\|r\| \|e\| \sin\alpha}{\|r\| \|e\| \cos\alpha + \|r\| \|r\|}$$

$$= \frac{\|r \times e\|}{r \cdot e + r \cdot r}$$

If we rotate $\bar{r}$ by ϕ around the axis $\frac{r \times e}{\|r \times e\|}$, r points toward Pd.

<u>Caveats</u> #1 This approach computes a <u>relative</u> rotation, e.g. ... the current direction of $\bar{r}$

Caveats #1 This approach computes a relative rotation, e.g. its based on the current direction of $\bar{r}$

→ multiply this rotation w/current rotation at each joint

→ can result in unatural rotations (such as twisting)

↖ to workaround, set joint rot. to a neutral rotation first (such as I)

#2: All calculations need to be in the same frame (aka coordinate system)

→ choose either global or local coordinates

↑ be carefull, a joint's local frame is its parent.

$\boxed{\text{Ex}}$ Conside the previous example where $l1 = 3$, $l2 = 2$, $p_d = (-3, \sqrt{7}, 0)^T$

(a) The desired length is still 4

(b) The angles ϕ, θ_{2z} are still the same, e.g. $\phi \sim 104°$, $\theta_{2z} \sim -75$

(c) Instead of computing $\gamma, \beta, \theta_{1z}$, we compute ϕ (from above) & axis $\frac{r \times e}{\|r \times e\|}$.

First, we need the position of p_3 in global coordinates after we set $\phi \sim 104°$ for elbow & $\theta_{2z} \sim -75$ for the shoulder.

$$P_3^0 = \begin{pmatrix} 3.517 \\ -1.931 \\ 0 \end{pmatrix}$$

aside, this comes from

$$P_3^0 = \left[\frac{R_1^0 R_2^1 d_3^2 + R_1^0 d_2^1 + d_1^0}{1}\right]$$

where $R_1^0 = I$

$R_2^1 = R_z(\theta_{2z}) = R_z(-75)$

$d_3^2 = \begin{pmatrix} l2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}$

$$d_3^2 = \begin{pmatrix} \hat{\ } \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$d_2^1 = \begin{pmatrix} l_1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix}$$

$$d_1^0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\bar{r} = P_3 - P_1 = \begin{pmatrix} 3.517 \\ -1.931 \\ 0 \end{pmatrix}$$

$$\bar{e} = P_d - P_3 = \begin{pmatrix} -3 \\ \sqrt{7} \\ 0 \end{pmatrix} - \begin{pmatrix} 3.517 \\ -1.931 \\ 0 \end{pmatrix} = \begin{pmatrix} -6.5 \\ 4.6 \\ 0 \end{pmatrix}$$

$$r \times e = \begin{pmatrix} 0 \\ 0 \\ 3.5 \end{pmatrix}, \quad r \cdot r = 16.106, \quad r \cdot e = -31.77$$

$$\left.\begin{aligned} &\Rightarrow \quad \phi = \operatorname{atan2}(\|r \times e\|, r \cdot r + r \cdot e) = 2.9211 \sim 167^\circ \\ &\text{axis} = (0, 0, 1)^T \end{aligned}\right] \text{set } R_{axis}(\phi) \text{ for } p'$$

IK Method #2: Cyclic Coordinate Descent (CCD)

Idea: Nudge each joint in a chain towards a goal position p_d

$\boxed{EX}$ Imagine a joint chain from the left hand to the root

Algorithm:

p = end effector's position (in global)

while $\|p_d - p\| >$ threshold and # iterations < maxIterations:

  for each joint in the chain from end effector to root

    "nudge" the joint towards p_d

    update p with the new end effector position

What is a nudge?

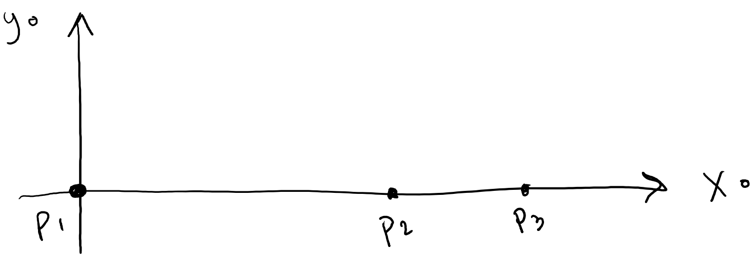
→ Use the tangent method to compute an angle ϕ & axis $\dfrac{r \times e}{\|r \times e\|}$, but only rotate a fraction

$\overline{\|r \times e\|}$

of ϕ, e.g.

nudge will be $\Delta\phi = c \;\operatorname{atan2}(\|r\times e\|, r\cdot r + r\cdot e)$
where $c \in [0,1]$ (typically ~0.1 is good)

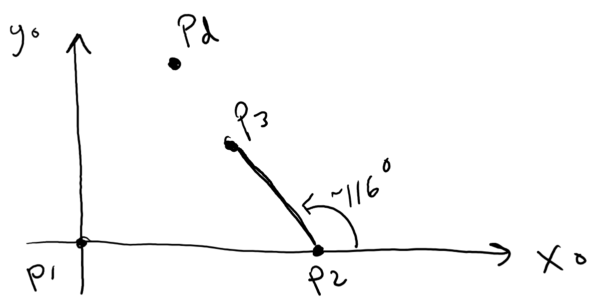
$\boxed{EX}$ 3 Link chain.



$R_1 = R_z(45)$
$R_2 = R_z(-45)$
$d_2^1 = (2,0,0)^T$
$d_3^2 = (1,0,0)^T$
$p_d = (1,2,0)^T$

Use CCD w/ nudge factor $c=1$ to move p_3 to p_d

<u>Step 1</u>: Compute ϕ + axis to rotate joint 2



$\bar{r} = p_3^0 - p_2^0 = (1,0,0)^T$

$\bar{e} = p_d^0 - p_3^0 = (-2,2,0)^T$

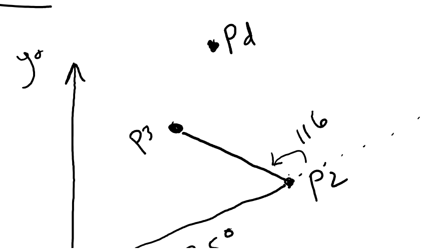
$\phi \sim 116°$

<u>Step 2</u>: What is the new global position of p_3 + p_2?

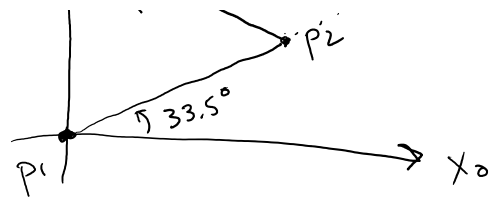
$p_2^0 = (2,0,0)^T$

$p_3^0 = (1.55, 0.89, 0)^T$

<u>Step 3</u>: Compute ϕ & axis to rotate joint 1



<u>note</u>: we miss $\Rightarrow$ this is why we "nudge" and iterate



and iterate

How to "point" any object towards a target

→ exs: steering a character by setting pos & ori of the root joint

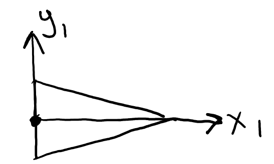
ex: looking at a target with the head

Idea: Use the fact that the columns of a rotation matrix correspond to the axes of the frame.

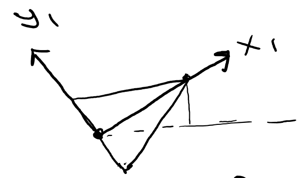
$\boxed{\text{EX}}$

$$R = \begin{pmatrix} 1/2 & -\sqrt{3}/2 & 0 \\ \sqrt{3}/2 & 1/2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

↑ x axis ↑ y axis ↑ z axis



In local frame x dir is always $(1,0,0)$, y is $(0,1,0)$, z is $(0,0,1)$



In global frame x dir is $R(1,0,0)^T$, y is $R(0,1,0)$ & z is $R(0,0,1)$

Insight: We can directly compute R so the forward direction points towards a target P_d.

$\boxed{\text{EX}}$ Suppose the forward direction is X, the global pos of our object is $(1,1,0)$ and $P_d = (4,-3,0)$

Our desired direction is $P_d - (1,1,0) = \begin{pmatrix} 4 \\ -3 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ -4 \\ 0 \end{pmatrix}$

Normalize the direction: $\left(\frac{3}{5}, -\frac{4}{5}, 0\right)^T$, and set as first column

$$\begin{pmatrix} 3/5 & \multirow{3}{*}{Y} & \multirow{3}{*}{Z} \\ -4/5 & & \\ 0 & & \end{pmatrix}$$

To get perpendicular Y & Z directions, assume Y is up and do:

$Z = X \text{ cross } (0,1,0)$

$$\begin{pmatrix} -4/5 & 1 & 2 \\ 0 & & \end{pmatrix}$$

do:

$Z = X$ cross $(0,1,0)$

$Y = Z$ cross X

where X is our desired forward direction

$$= \begin{pmatrix} 3/5 & +4/5 & 0 \\ -4/5 & 3/5 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

In general, if Z corresponds to the fwd direction of a character (or object), the corresponding rotation R^{global}_{local} is

$$Z = \text{target} - \text{characterPos}$$

$$X = Up \times Z$$, where $Up = (0,1,0)$ in our base code

$$Y = Z \times X$$

where X, Y, Z are computed in global coordinates

$$R^{global}_{local} = \begin{pmatrix} \dfrac{X}{\|X\|} & \dfrac{Y}{\|Y\|} & \dfrac{Z}{\|Z\|} \end{pmatrix}$$