Inverse Kinematics for Computer Animation

Roman Berka

http://vyuka.iim.cz/a4m39mma:a4m39mma

How to make them moving?

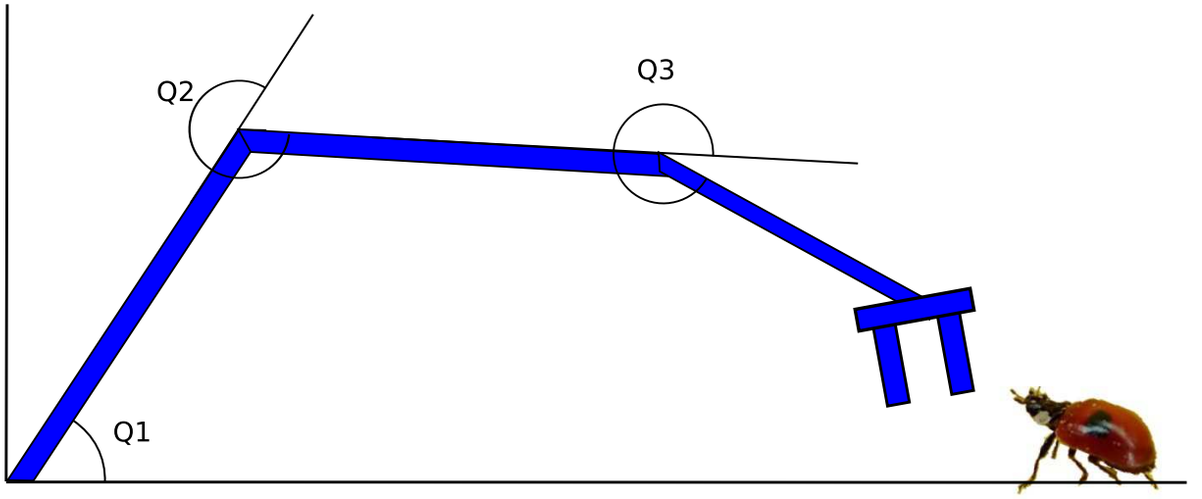
Why we solve the problem of IK?

- analytic solution is often impossible
  - too much solutions (large solution space)
  - no solution (empty solution space)
- we need a numeric solution (can be expensive)
- we are looking for a robust and fast solution.

Talk Overview

1. IK problem specification
2. Jacobian inversion method
3. Jacobian transposition method
4. Cyclic Coordinate Descent method
5. Comparison

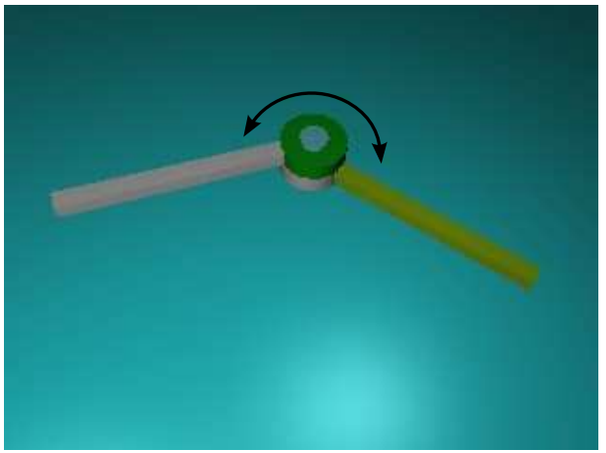
Rigid parts (links) connected with joints



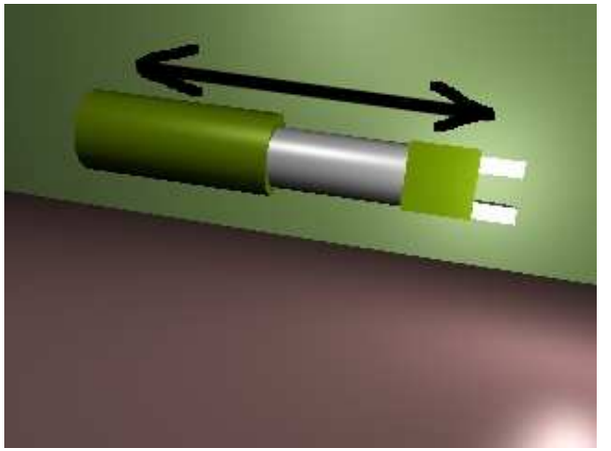
We will assume some limits:

- constant size of each link.
- base of whole structure is located in O[0,0,0]
- open kinematics chain only (e.g. a human arm without loops)

Joint Basic Types



revolute joint (q_i)



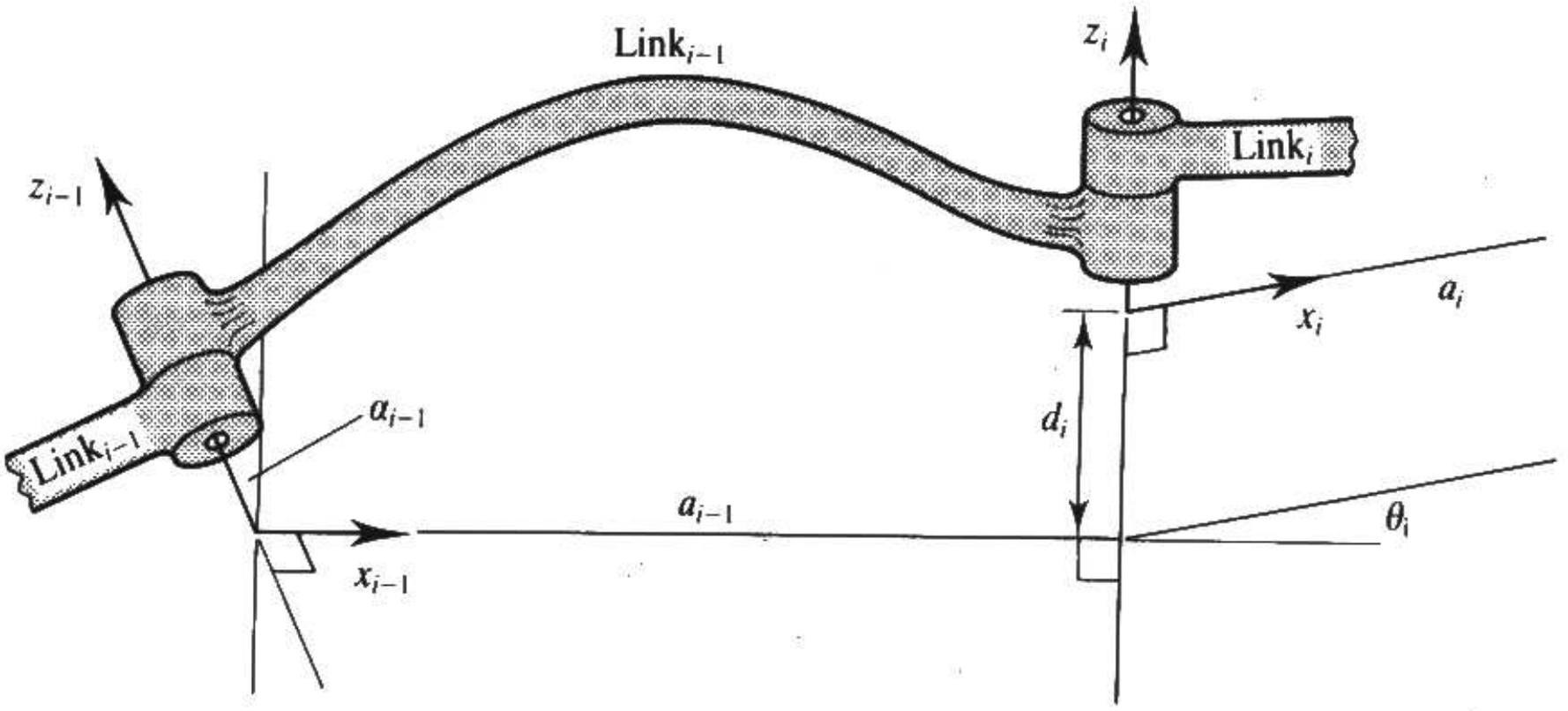
prismatic joint (d_i)

Any other types:

- can be more complex
- can be modeled as combination of 1DOF joints

DH Notation

Denavit-Hartenberg (1955!)



link twist - α_{i-1}, link length - a_{i-1},
link offset - d_i, joint angle - Θ_i

- let Θ be a configuration expressed in state space using both q_i and d_i e.g. $\Theta = (\theta_1, \theta_2, \theta_3, \ldots, \theta_n)$ for revolute joints.
- and X is position of the end effector expressed in Cartesian space.
- X is expressed with 3DOF(position) or 6DOF (position + orientation).
- Then the relation between Θ and X is given as:

$$\boxed{X = f(\Theta)}$$

What is the funcion f?

- each link L_i is positioned in coordinates of its predecessor L_{i-1}.
- the coordinate system of link L_i is denoted as $frame\ \{i\}$.
- the transformation between frames is given by four parameters: α_i, a_i, d_i, and Θ_i.

then the relation between frames $\{i\}$ and $\{i-1\}$ is given by concatenation of four transformations:

$$
{}^{i-1}T_i = \begin{bmatrix} R(\Theta_i)^z & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ (0,0,d_i) & 1 \end{bmatrix} \cdot
$$

$$
\cdot \begin{bmatrix} 1 & 0 \\ (a_{i-1},0,0) & 1 \end{bmatrix} \cdot \begin{bmatrix} R(\alpha_{i-1})^x & 0 \\ 0 & 1 \end{bmatrix}
$$

Representation of the function f

DCGI

- ${}^{i}x_i$ viewed from the frame $\{i-1\}$ is then expressed as:

$$ {}^{i-1}x_i = {}^{i}\, x_i . {}^{i-1}T_i $$

- ...and ${}^{i}x_i$ viewed from the frame $\{0\}$ is then expressed as:

$$ {}^{0}x_i = {}^{i}\, x_i . {}^{0}T_i = {}^{i}\, x_i . \prod_{j=i}^{1} {}^{j-1}T_j $$

- $X = f(\Theta) \longleftrightarrow X = ({}^{i}x_i . \prod_{j=n}^{1} {}^{j-1}T_j)(\Theta)$

Let's go to opposite direction!

- forward: configuration $\Longrightarrow$ EE position $\quad X = f(\Theta)$
- inverse: EE position $\Longrightarrow$ configuration $\quad \Theta = f^{-1}(X)$
- What is f^{-1}?
- f is non-linear owing to $\cos$ and $\sin$ functions
- we have to find any linear approximation for f^{-1}

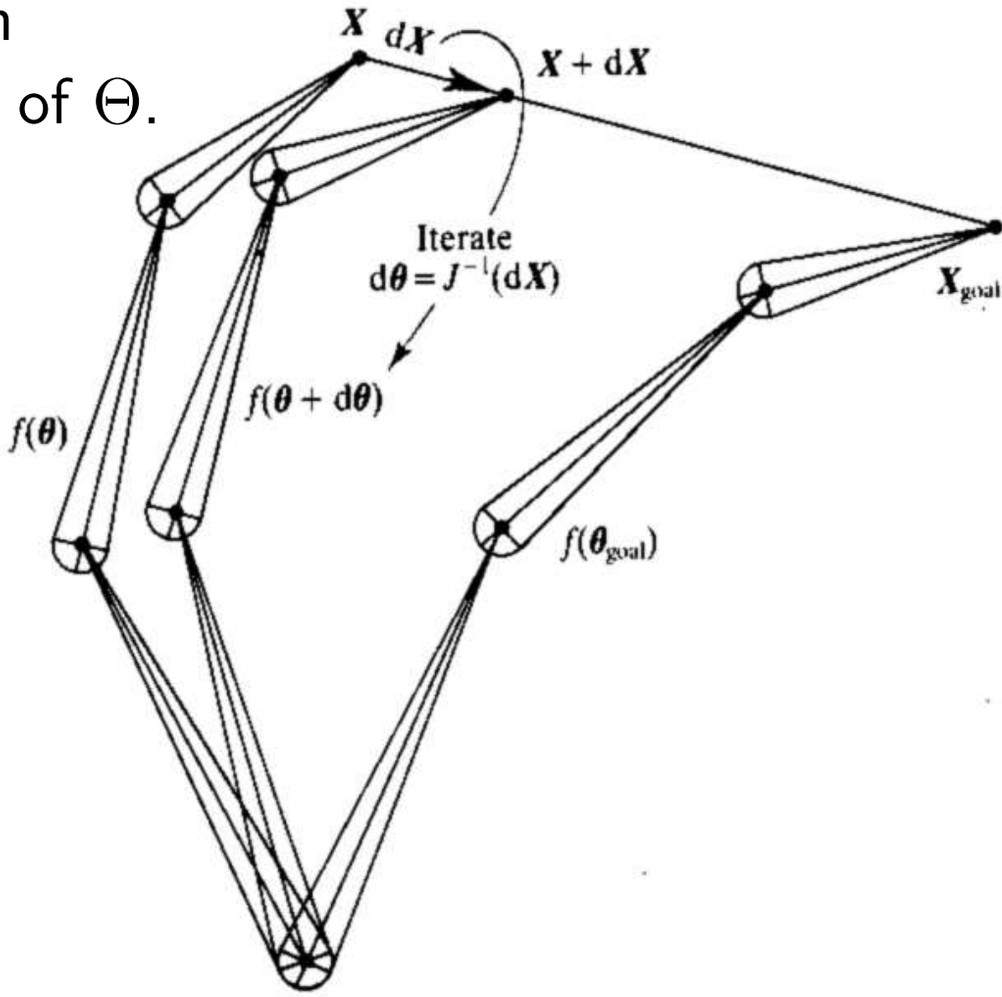
- forward kinematics $X = f(\Theta) = (f_1(\Theta), f_2(\Theta), \ldots, f_6(\Theta))$
- The linear approximations can be given using the $6 \times n$ matrix Jacobian

$$J(\Theta) = \left[\frac{\partial f_i}{\partial q_j}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial q_1} & \cdots & \frac{\partial f_6}{\partial q_n} \end{bmatrix}$$

- the partial derivatives must be evaluated in a small surrounding of Θ to match local linear approximation of f^{-1}.

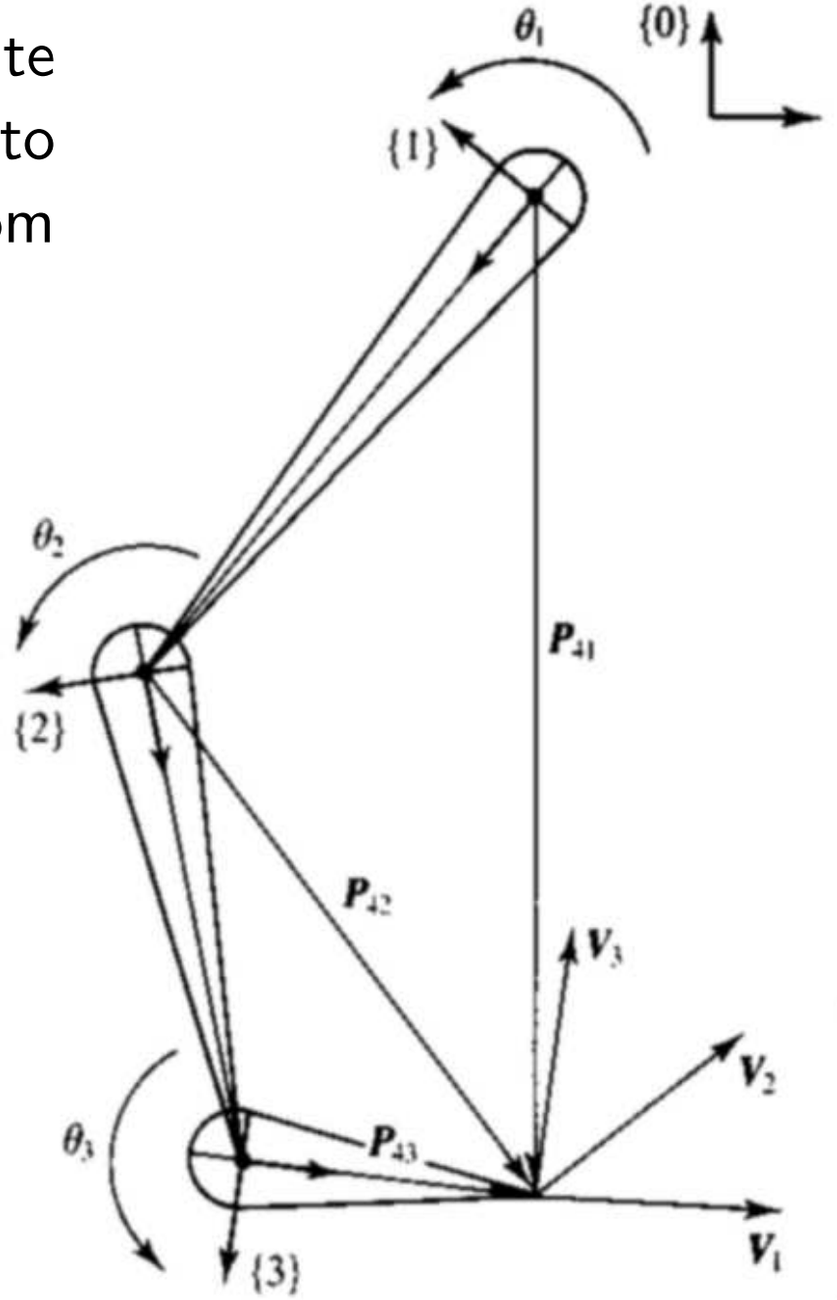
The Jacobian

- $f(\hat{\Theta}) + [J(\hat{\Theta})](\Theta - \hat{\Theta})$ is the 1^{st} order Taylor approximation of f at $\hat{\Theta}$[1].
- The Jacobian tell us how position of EE changes when tuning parts of Θ.
- $\Delta X = J(\Theta)\Delta\Theta \Rightarrow$
  $\Rightarrow \Delta\Theta = J^{-1}(\Theta)\Delta X$



[1] $f(x) = \sum_{i=0}^{n} \frac{f^{(i)}(a)}{i!}(x-a)^i + R_n$

Now, the problem is to compute the Jacobian. Let's go to extract necessary information from transformation matrices!



- All we need is to express velocities (linear $\mathbf{v}$ and angular ω) of EE in global space (frame $\{0\}$) based on local parameters of links.
- we can write $\mathbf{v}_{i,0} = \omega_{i,i-1} \times (\mathbf{P}_n - \mathbf{P}_i)$
- in fact, the cross product originates from

  $$\Omega = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$ and

  $$\Omega.(\mathbf{P}_n - \mathbf{P}_i) \equiv (\omega_x, \omega_y, \omega_z) \times (\mathbf{P}_n - \mathbf{P}_i)$$
- then the resulting linear velocity is $\mathbf{v}_{n,0} = \sum_{i=1}^{n-1} \omega_{i,i-1} \times (\mathbf{P}_n - \mathbf{P}_i)$
- the angular velocity of EE is optional - $\omega_{n,0} = \mathbf{P}_{n,0} \times \mathbf{v}_{n,0}$

- expression of $\mathbf{P}_i$ and in frame $\{0\}$ we can obtain as $P_{i,0} = (0, 0, 0, 1).{}^0T_i$ which is the fourth row of ${}^0T_i$
- for $\omega_{i,0}$ we need just only transformed rotational axis $a_{zi} = (0, 0, 1, 1).{}^0T_i$ which is just first three elements of third row of ${}^0T_i$

Finally we can construct Jacobian so that one column i looks like:

$$J_i(\Theta) = \begin{bmatrix} [\mathbf{axis}_{zi} \times (\mathbf{P}_n - \mathbf{P}_i)]^T \\ [\mathbf{axis}_{zi}]^T \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{zi} \\ \mathbf{a}_{zi} \end{bmatrix}$$

where ${}^0T_i = \begin{bmatrix} \mathbf{axis}_{xi} & 0 \\ \mathbf{axis}_{yi} & 0 \\ \mathbf{axis}_{zi} & 0 \\ \mathbf{P}_i & 1 \end{bmatrix}$

The final equation:

$$\begin{bmatrix} \mathbf{v}_{n,0} \\ \omega_{n,0} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{z1} & \dots & \mathbf{b}_{zi} & \dots & \mathbf{b}_{zn-1} \\ \mathbf{a}_{z1} & \dots & \mathbf{a}_{zi} & \dots & \mathbf{a}_{zn-1} \end{bmatrix} \begin{bmatrix} \Delta q_1 \\ \dots \\ \Delta q_i \\ \dots \\ \Delta q_{n-1} \end{bmatrix}$$

- Now we need to solve $\Delta\Theta = J^{-1}(\Theta)\Delta X$.
- The J is typically non-invertible because DOFs for EE (3 or 6) and for whole chain are usually different.
- How to invert it?
- Using generalized pseudo-inverse J^+ which gives an unique solution and has the following properties:
  - $JJ^+J = J \qquad J^+JJ^+ = J^+$
  - $(J^+J)^T = J^+J \qquad (JJ^+)^T = JJ^+$

- the solution based on J^+ is possible source of errors
- $dX = X_{goal} - X$ may exceed any threshold value.
- iteration is needed.
- so we minimize $Err = ||J^+(\Theta)\Delta\Theta - X_{goal}||$
- so if $Err > \varepsilon$ we compute $X_i = \frac{X+X_{goal}}{2}$ and iterate first over interval X, X_i.

Jacobian Transpose Method

- avoids inversion of Jacobian
- is based on the idea of virtual works
  $Work = force \times distance$ $\quad$ $Work = torque \times angle$
- so $F.\Delta X = \tau.\Delta\Theta$
  or $F^T.\Delta X = \tau^T.\Delta\Theta$
- we know $\Delta X = J.\Delta\Theta$
- thus $F^T.J\Delta\Theta = \tau^T.\Delta\Theta$
- then $F^T.J = \tau^T \Rightarrow J^T.F = \tau$

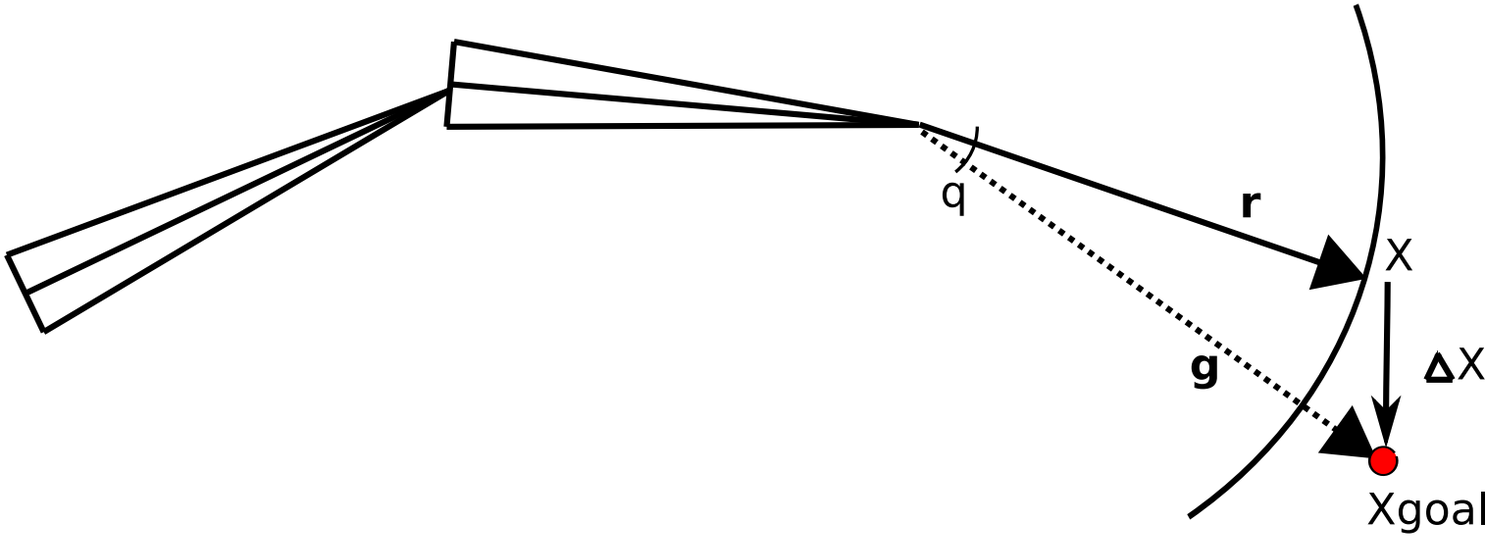
Jacobian Transpose

$\tau = J^T.F$ $\longrightarrow$ $\Delta\Theta = J^T.\Delta X$

virt. force equation — *compare with formal equation*

- we use the distance $||X_{goal} - X||$ as a Force that pulls the EE.
- using a scaling factor λ we can iterate $\Delta\Theta^{(i+1)} = \lambda J^T.\Delta X^{(i)}$
- λ can be interpreted as a time-step Δt

CCD



- based on simple idea when changing only 1DOF per iteration
- the main problem is to find the angle q
- finding the q can be defined as maximization task
- ...because ΔX is minimized when $\mathbf{g}.\mathbf{r}$ is maximized

Comparing the Three Methods

- Jacobian inversion
  - \+ J^+ gives solutions having minimal form in each step
  - \+ faster converge then J^T
  - \- singularities of J
  - \- complicated implementation
- Jacobian Transpose
  - \+ cheaper evaluation
  - \+ no singularities
  - \- scaling problem

Comparing the Three Methods

- CCD
  - + no singularities
  - + cheap evaluation
  - + simple to implement
  - - does not necessary lead to smooth solution
  - - can give odd solution when deltas in step are not limited

Some Useful Tools

- IK solvers - usually part of a modeling software (usually commercial)
- e.g. Alias Maya - open C++ architecture
- Internet resources - sometimes free code can be found
  - Java3D – http://www.brockeng.com/VMech/IK/IKSG.htm
  - Java applet – http://www.nbb.cornell.edu/neurobio/land/OldStudentProjects/cs490-96to97/hoffman/
  - Example in Processing – http://www.openprocessing.org/visuals/?visualID=12368

References

- H. Watt, A. and M. Watt. *Advanced Animation and Rendering Techniques*. Addison Wesley, 1992.
- K. W. Chin. Closed-Form and Generalized Inverse Kinematic Solutions for Animating the Human Articulated Structure. Technical report, Curtin University of Technology, 1996.
- C. Welman. Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation. Master's thesis, Simon Fraser University, September 1993.
- J. J. Craig. *Adaptive Control of Mechanical Manipulators*. Addison-Wesley, 1988. ISBN 0-201-10-490-3.