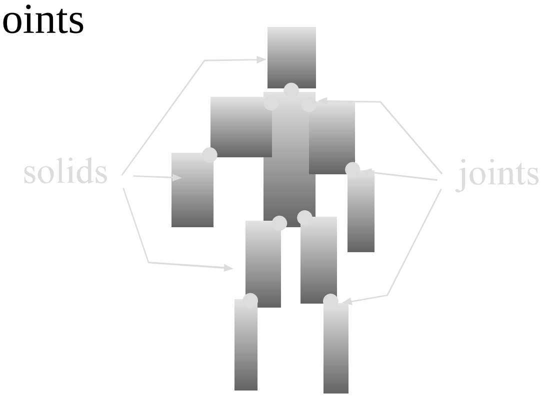
Kinematic animation of articulated bodies

- Joints
- Kinematic graph
- Forward kinematics
- Inverse kinematics

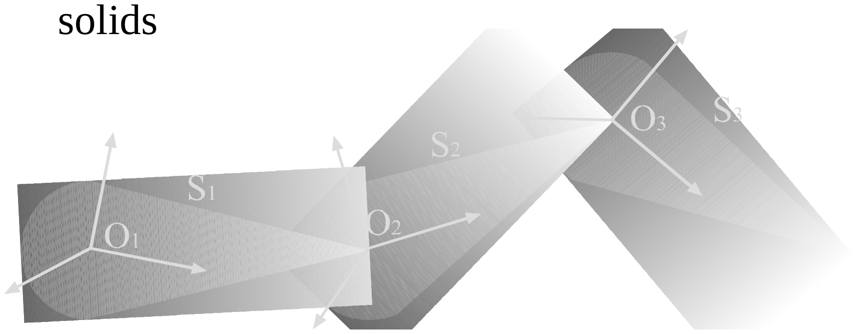
Articulated bodies

- Articulated bodies are composed of solids and joints



Joints

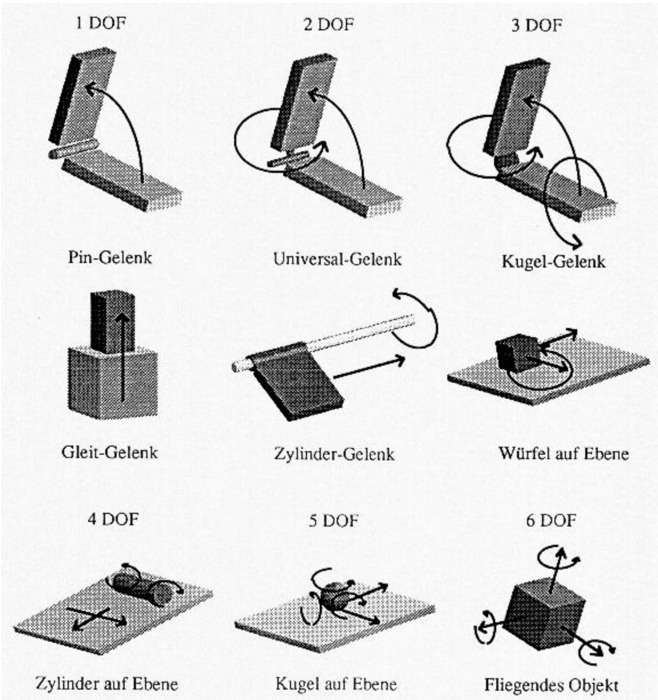
- Joints restrict the relative motion of solids
- Joints are used to create hierarchies of solids



Degrees of freedom (DOF)

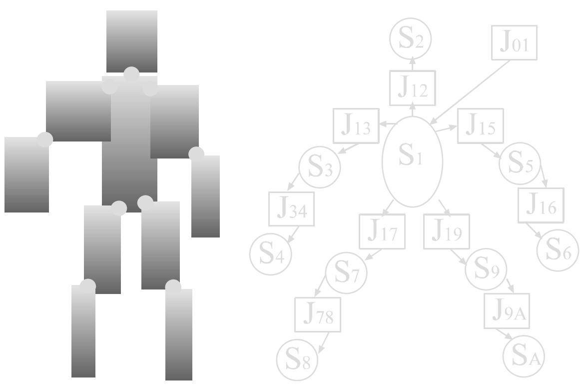
- The dofs define the independent relative motions allowed
- Each joint can include a combination of:
  - 3 translation dofs
  - 3 rotation dofs

degrees of freedom (DOF)



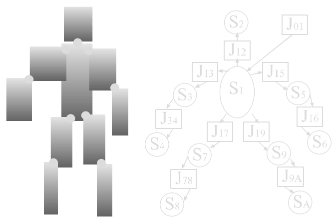
Kinematic graph

- The kinematic graph defines the structure of the articulated body



Data structures

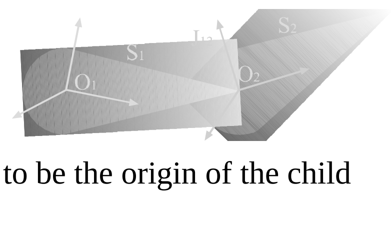
- Kinematic graph:
  - list of root joints (one/articulated body)
  - nodes: joints and solids
- Solid:
  - parent joint
  - list of child joints
  - transform wrt world coordinates



Data structures (continued)

- Joint:
  - parent solid
  - child solid
  - transform wrt parent

    We chose the joint frame to be the origin of the child solid
  - dofs (ex: translation i, rotation j,k)
  - state variables (one/dof)



Recursive computations

- Solid transforms: trans(root)

  – trans(solid S):

  for all children joints j
  trans(j)

  – trans(joint J):

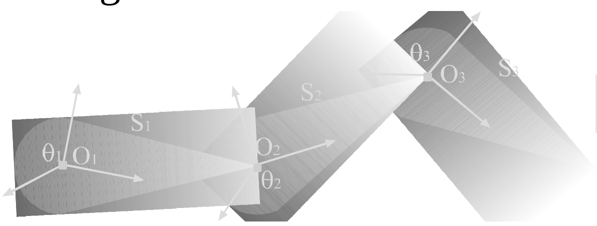
  $$^{\text{parent(J)}}\mathbf{T}_{\text{child(J)}} = \mathbf{T}(dof\ )$$

  $$^{0}\mathbf{T}_{\text{child(J)}} = {}^{0}\mathbf{T}_{\text{parent(J)}}\ {}^{\text{parent(J)}}\mathbf{T}_{\text{child(J)}}$$

  trans(child)

State vector

- The state vector gathers the displacements along all the dofs of the scene



$$\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^T$$

- An animation is a path θ*(t)* in the state space

Forward kinematics

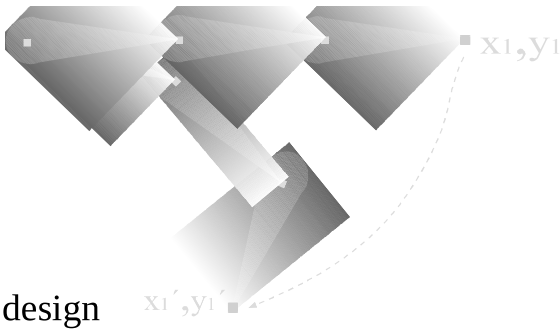
- Some artifacts of point animation are removed

Forward kinematics (continued)

- However, forward kinematics has to be applied carefully

Inverse kinematics

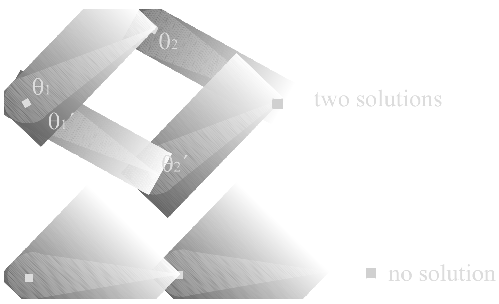
- Inverse kinematics allows us to apply constraints



- Applications:
  - interactive pose design
  - applying goals
  - maintaining geometric relations

Geometric equations

- Nonlinearity:
  - various number of solutions
  - difficult to solve



Linearized equations

- Newton´s method for nonlinear root finding:

  while(error > precision)

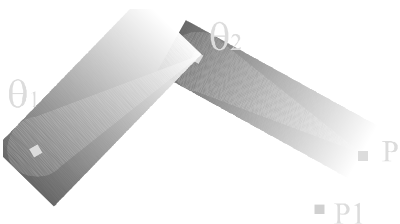
  approximate the function by its derivative

  solve

Example

- Point-to-point constraint

  $\mathbf{P} = \mathbf{P}_1$



- Solve $\dfrac{\mathbf{dP}}{\mathbf{d\theta}}\Delta\theta = \mathbf{P}_1 - \mathbf{P}$

- Update $\theta = \theta + \Delta\theta$

- Check state $(\mathbf{P}_1 - \mathbf{P})(\theta)$

Constraints

- Constraints can be expressed as vectors
- Each entry (scalar constraint) is an independent element of the consraint
- example: point-to-point

$$\mathbf{P}_1 - \mathbf{P} = \Delta\mathbf{P} = \mathbf{0}$$

$$\mathbf{g} = \begin{pmatrix} \Delta\mathbf{P.i} \\ \Delta\mathbf{P.j} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

Jacobian

- The jacobian matrix J of a constraint relates its value to the state variables

$$\mathbf{J} = \frac{\mathbf{dg}}{\mathbf{d\theta}}$$

- Each row of J is the gradient of a scalar constraint

$$\mathbf{J} = \begin{bmatrix} \partial g_1 / \partial\theta_1 & \ldots & \partial g_1 / \partial\theta_n \\ \ldots & \ldots & \ldots \\ \partial g_m / \partial\theta_1 & \ldots & \partial g_m / \partial\theta_1 \end{bmatrix}$$

Jacobian

- J provides good approximations for small displacements

$$\mathbf{g}(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx \mathbf{g}(\boldsymbol{\theta}) + \mathbf{J}\Delta\theta$$

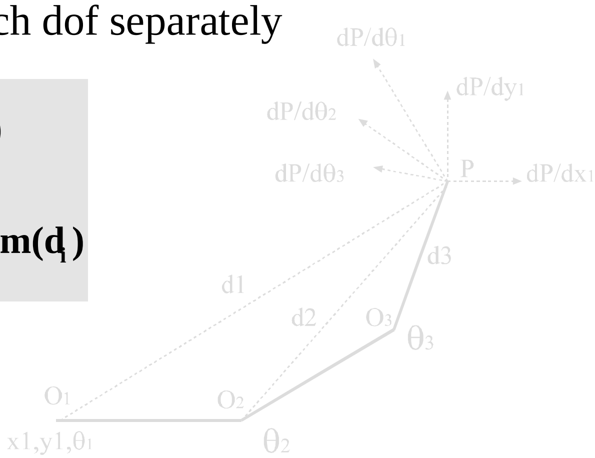
- J is not necessarily square
  – m scalar constraints
  – n unknowns
  -> J(m,n)

Computation of the Jacobian matrix

- Consider each dof separately

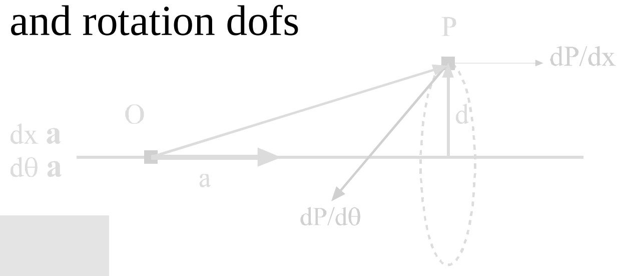
$$\frac{\mathbf{dP}}{\mathbf{d}x_i} = \mathbf{axis}(x_i)$$

$$\frac{\mathbf{dP}}{\mathbf{d}\theta_i} = \|\mathbf{d_i}\|\,\mathbf{norm}(\mathbf{d_i})$$



Relations in 3D

- Translation and rotation dofs



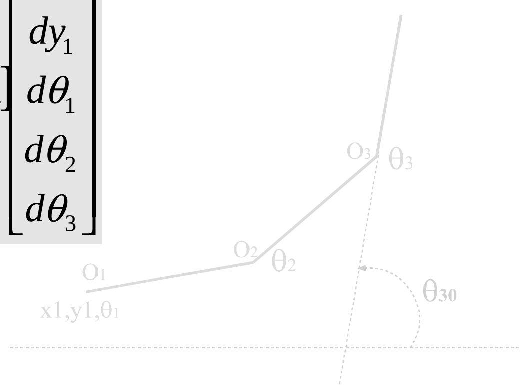
$$\frac{\mathbf{dP}}{\mathbf{d}x} = \mathbf{a}$$

$$\frac{\mathbf{dP}}{\mathbf{d}\theta} = \mathbf{a} \times \mathbf{d} = \mathbf{a} \times \mathbf{OP}$$

Orientations

- Orientations depend only on rotations

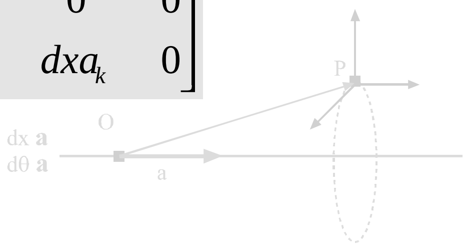
$$d\theta_{30} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} dx_1 \\ dy_1 \\ d\theta_1 \\ d\theta_2 \\ d\theta_3 \end{bmatrix}$$



Transforms in 3D

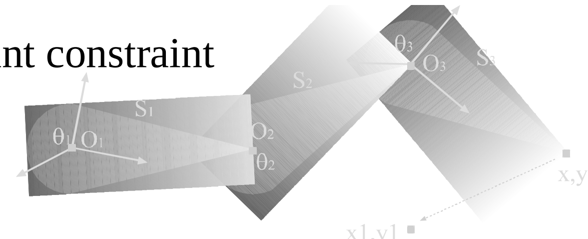
- Derivative of the transform matrix

$$d\mathbf{T}=\begin{bmatrix} 0 & d\theta a_k & -d\theta a_j & 0 \\ -d\theta a_k & 0 & d\theta a_i & 0 \\ d\theta a_j & -d\theta a_i & 0 & 0 \\ dxa_i & dxa_j & dxa_k & 0 \end{bmatrix}$$



Example 1

- point-to-point constraint



$$\begin{bmatrix} \frac{\partial \mathbf{P}}{\partial \theta_1}.\mathbf{i} & \frac{\partial \mathbf{P}}{\partial \theta_2}.\mathbf{i} & \frac{\partial \mathbf{P}}{\partial \theta_3}.\mathbf{i} \\ \frac{\partial \mathbf{P}}{\partial \theta_1}.\mathbf{j} & \frac{\partial \mathbf{P}}{\partial \theta_2}.\mathbf{j} & \frac{\partial \mathbf{P}}{\partial \theta_3}.\mathbf{j} \end{bmatrix}\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix}=\begin{bmatrix} x_1-x \\ y_1-y \end{bmatrix}$$

Example 2

- 2-dof body

$$\frac{\partial \mathbf{P}}{\partial x} = \mathbf{i}$$

$$\frac{\partial \mathbf{P}}{\partial \theta} = \mathbf{k} \times \mathbf{O_1 P}$$

$$\mathbf{\Delta P} = \frac{\partial \mathbf{P}}{\partial x}\Delta x + \frac{\partial \mathbf{P}}{\partial \theta}\Delta \theta$$

Example 2 (continued)

- Point-point constraint:
  - derive the equations

$$\mathbf{\Delta P} = \frac{\partial \mathbf{P}}{\partial x}\Delta x + \frac{\partial \mathbf{P}}{\partial \theta}\Delta \theta = \mathbf{g} = \mathbf{P_1} - \mathbf{P}$$

$$\begin{bmatrix} \frac{\partial \mathbf{P}}{\partial x}.\mathbf{i} & \frac{\partial \mathbf{P}}{\partial \theta}.\mathbf{i} \\ \frac{\partial \mathbf{P}}{\partial x}.\mathbf{j} & \frac{\partial \mathbf{P}}{\partial \theta}.\mathbf{j} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \theta \end{bmatrix} = \begin{bmatrix} (\mathbf{P_1} - \mathbf{P}).\mathbf{i} \\ (\mathbf{P_1} - \mathbf{P}).\mathbf{j} \end{bmatrix}$$

$$\begin{bmatrix} 1 & (\mathbf{k} \times O_1 P).\mathbf{i} \\ 0 & (\mathbf{k} \times O_1 P).\mathbf{j} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \theta \end{bmatrix} = \begin{bmatrix} x_1 - x \\ y_1 - y \end{bmatrix}$$

Example 2 (continued)

- Point-line constraint

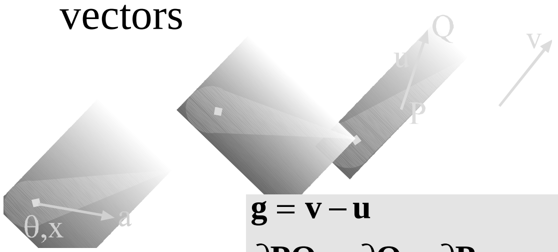
$$\Delta\mathbf{P}.\mathbf{n} = (\frac{\partial\mathbf{P}}{\partial x}\Delta x + \frac{\partial\mathbf{P}}{\partial\theta}\Delta\theta).\mathbf{n} = -d = (\mathbf{P_1} - \mathbf{P}).\mathbf{n}$$

$$\begin{bmatrix}\frac{\partial\mathbf{P}}{\partial x}.\mathbf{n} & \frac{\partial\mathbf{P}}{\partial\theta}.\mathbf{n}\end{bmatrix}\begin{bmatrix}\Delta x \\ \Delta\theta\end{bmatrix} = d$$

$$\begin{bmatrix}\mathbf{i}.\mathbf{n} & (\mathbf{k}\times O_1 P).\mathbf{n}\end{bmatrix}\begin{bmatrix}\Delta x \\ \Delta\theta\end{bmatrix} = -d$$

Orientation constraints

- Orientation can be constrained by aligning vectors



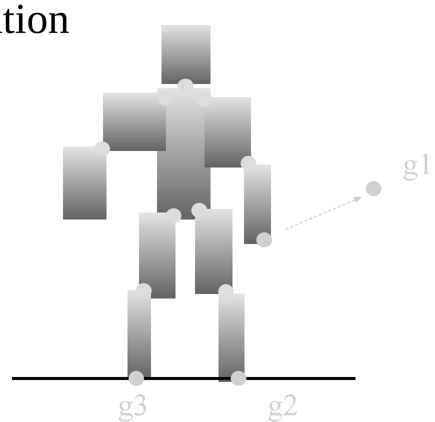
$$\mathbf{g} = \mathbf{v} - \mathbf{u}$$

$$\frac{\partial\mathbf{PQ}}{\partial x} = \frac{\partial\mathbf{Q}}{\partial x} - \frac{\partial\mathbf{P}}{\partial x} = \mathbf{a} - \mathbf{a} = \mathbf{0}$$

$$\frac{\partial\mathbf{PQ}}{\partial\theta} = \frac{\partial\mathbf{Q}}{\partial\theta} - \frac{\partial\mathbf{P}}{\partial\theta} = \mathbf{a}\times\mathbf{O_iB} - \mathbf{a}\times\mathbf{O_iA} = \mathbf{a}\times\mathbf{AB}$$

Multiple constraints

- We gather all the constraints in one equation



$$\begin{bmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \mathbf{J}_3 \end{bmatrix} \Delta\boldsymbol{\theta} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \mathbf{g}_3 \end{bmatrix}$$

Inversion of the Jacobian matrix

- If $J_{(m,n)}$ is not square, use the pseudoinverse
  - full rank matrices:

$$m > n : \mathbf{J}^{+} = (\mathbf{J}^{T}\mathbf{J})^{-1}\mathbf{J}^{T}$$
$$m < n : \mathbf{J}^{+} = \mathbf{J}^{T}(\mathbf{J}\mathbf{J}^{T})^{-1}$$

  - rank deficient matrices: use SVD or other methods

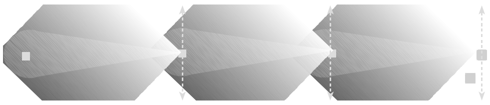
Solution of the equation system

- Compute the unknowns $\Delta\theta$ (translations and rotations) to meet the constraints Δx

$$J\,\Delta\theta = \Delta x$$

$$\Delta\theta = J^{+}\Delta x$$

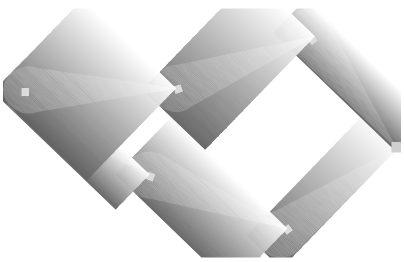
- Degenerate matrices require special care



$$\min_{\Delta\theta}(J\Delta\theta - \Delta x)^2 + \lambda\Delta\theta^2$$

Global degrees of freedom

- if $m<n$, there remain some global degrees of freedom



The null space

- The null space of J is the set of vectors wich have no influence on the constraints

$$\theta \in nullspace(J) \Leftrightarrow J\theta = 0$$

- The pseudoinverse provides an operator which projects any vector to the null space of J.

$$\mathrm{J}\,\Delta\theta = \Delta\mathrm{x}$$

$$\Delta\theta = \mathrm{J}^{+}\Delta\mathrm{x} + (\mathrm{I} - \mathrm{J}^{+}\mathrm{J})\mathrm{z} \qquad \forall \mathrm{z}$$

Utility of the null space

- The null space can be used to reach secondary goals

$$\Delta\boldsymbol{\theta} = \mathrm{J}^{+}\Delta\mathbf{x} + (\mathrm{I} - \mathrm{J}^{+}\mathrm{J})\mathbf{z}$$

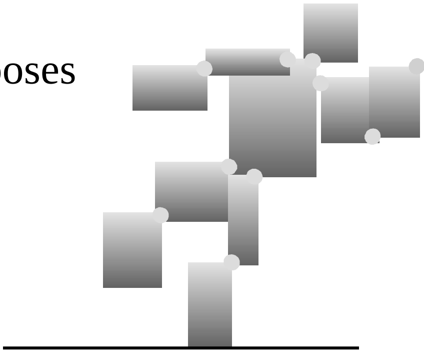
$$\min_{\mathrm{z}} f(\boldsymbol{\theta})$$

- Example: comfortable positions

$$f(\boldsymbol{\theta}) = \sum_i (\theta_{comfort}(i) - \theta(i))^2$$

Application to pose optimization

- Avoid unrealistic poses

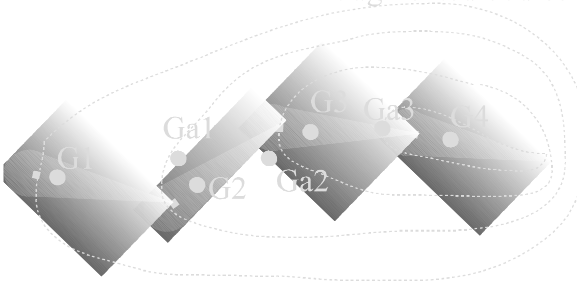


- Maintain the center of mass above the feet

$$G = \sum_i m_i G_i$$

Augmented body

- The “augmented body” (Boulic94) corresponds to the mass moved by a given joint.



$$G = \sum_i m_i G_i$$

$$\frac{\partial G}{\partial \theta_i} = (\sum_i^n m_i)\frac{\partial Ga_i}{\partial \theta_i}$$

Optimization of the mass center

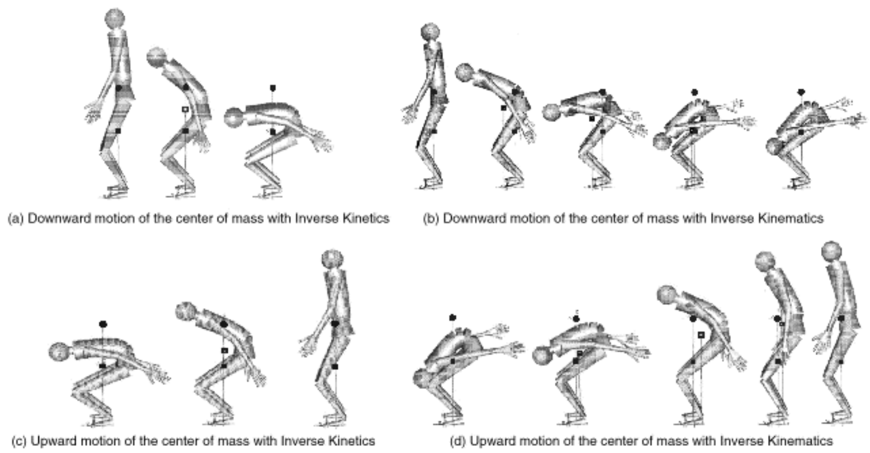
- The null space of the kinematic constraints is used to optimize the position of the mass center

$$\Delta\boldsymbol{\theta} = \mathrm{J}^{+}\Delta\mathbf{x} + (\mathrm{I} - \mathrm{J}^{+}\mathrm{J})\mathbf{z}$$

$$\min_{\mathrm{z}} f(G(\boldsymbol{\theta}))$$

- The derivative $\delta G/\delta\theta$ is used in the minimization process

Examples



(a) Downward motion of the center of mass with Inverse Kinetics

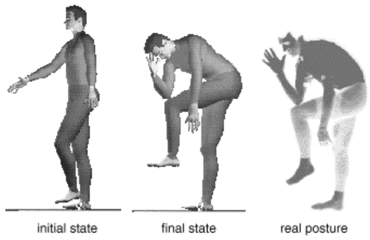
(b) Downward motion of the center of mass with Inverse Kinematics

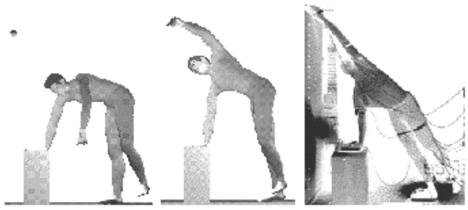
(c) Upward motion of the center of mass with Inverse Kinetics

(d) Upward motion of the center of mass with Inverse Kinematics

Applications

- Single support
- Double support





Summary of inverse kinematics

- Inverse kinematics allows us to define goals
- The Jacobian matrix J relates the constrained values to the control variables
- The pseudoinverse allows us to solve the equation system
- The null space of J allows us to reach secondary goals (but which ones?)
- Due to linearization or degeneracies, the process may be instable and solved iteratively

References

- Press, Teukolski, Vetterling, Flannery, *Numerical recipes in C*, Cambridge University press
- Zhao, Badler, *Inverse kinematics positioning using nonlinear programming for higly articulated figures*, ACM Transactions on Graphics, 13(4), 1994
- Boulic, R., Mas-Sanso, R., Thalmann, D., *Complex character positionning based on a compatible flow model of multiple supports*, IEEE Transactions on Visualization and Computer Graphics, vol 3, no 3, July-September 1997