Motion Control Methods for Skeleton

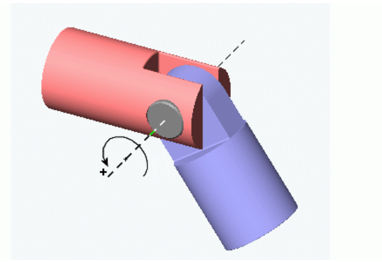
Daniel Thalmann

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Cagliari, May 2008

Animation of articulated bodies

- Characters, humans, animals, robots.
- Characterized by hierarchical structure: skeleton.
- Skeleton: set of interconnected segments corresponding to members and joints.
- Joint: intersection of two segments.
- Angle between two segments: joint angle.
- Animation: joint can have up to 3 joint angles: flexion, pivot, twisting.

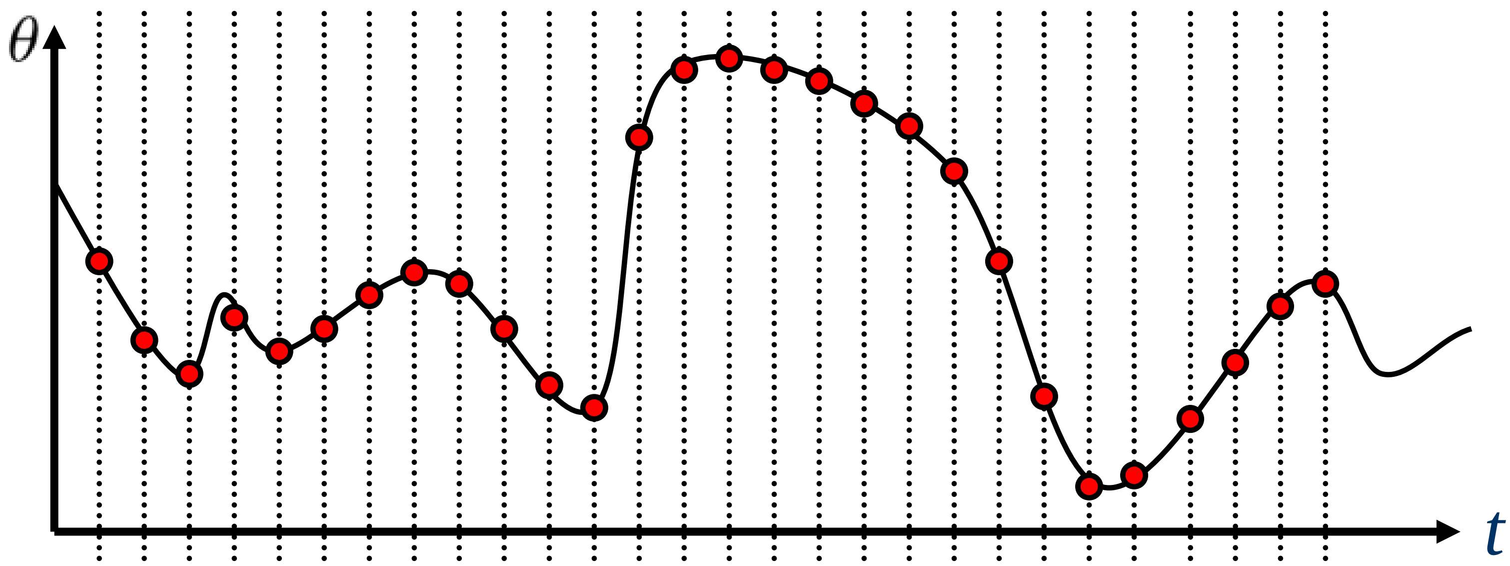


Skeletal Animation

- Skeletal motion is function of time

$$\theta = f(t)$$

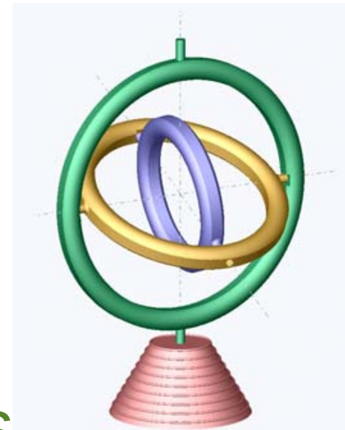
- Representing this function

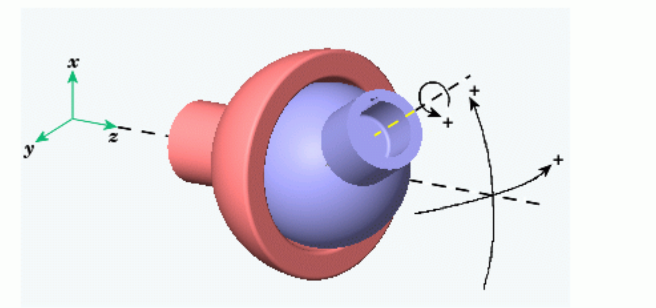


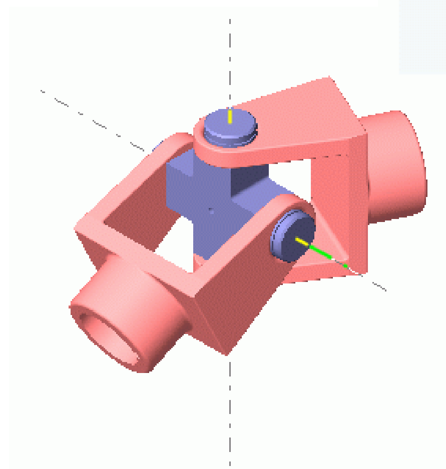
$$\theta_1 = f(t_1)$$
$$\theta_2 = f(t_2)$$
$$\theta_3 = f(t_3)$$
$$\cdots$$
$$\theta_n = f(t_n)$$

Complex Joints

- 3 DOF joints
  - Gimbal
  - Spherical (doesn't possess singularity)
- 2 DOF joints
  - Universal

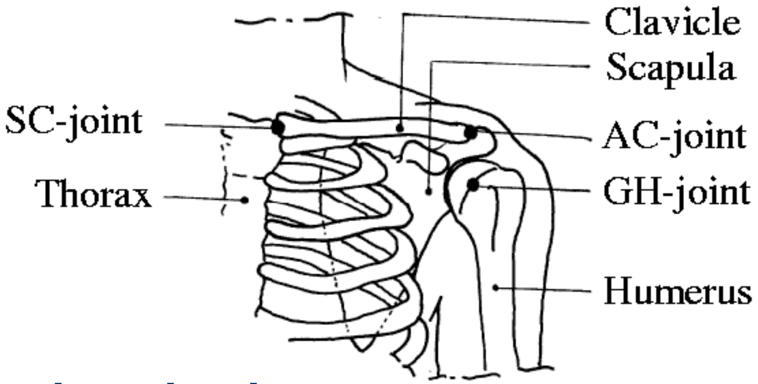




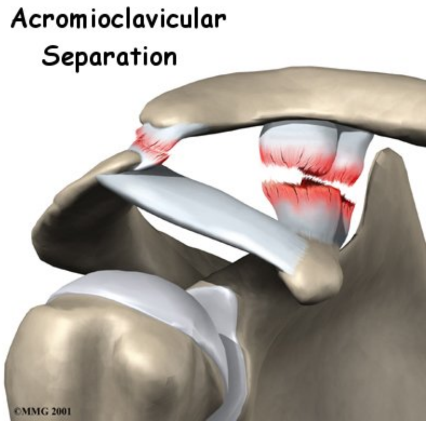


Example: Joints in the Shoulder Complex

- SC-joint: 3 DoF
- AC-joint: 3 DoF
- ST-"joint": 4 DoF
- GH-joint: 3 DoF

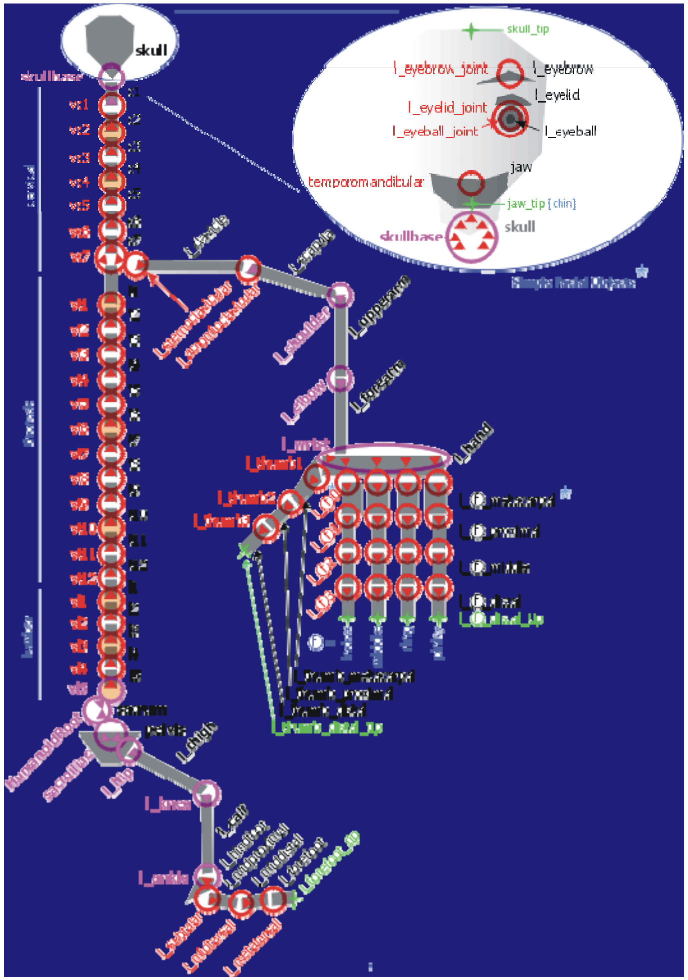


- SC + AC + ST: closed kinematic chain
  - 4 DoF between Thorax and Glenoid
  - 2 translations
  - 2 rotations
    - Governed by GH
- Resulting DoF
  - 2 prismatic joints: 2 DoF
  - GH-joint: 3 DoF



Hierarchy Representation

- Model bodies (links) as nodes of tree
- All body frames local (relative to parent)
    - Transformations affecting any node affect all its children
    - H-ANIM



Overview

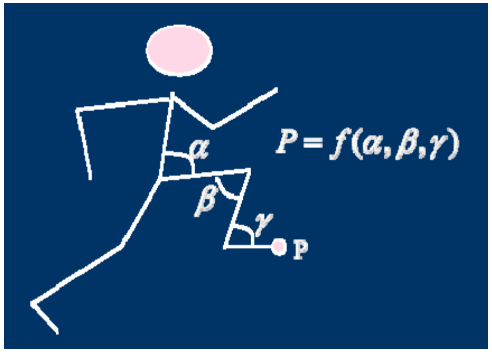
- **Kinematics**
  - Consider only motion
  - Determined by positions, velocities, accelerations
- **Dynamics**
  - Consider underlying forces
  - Compute motion from initial conditions and physics

Motion capture animation

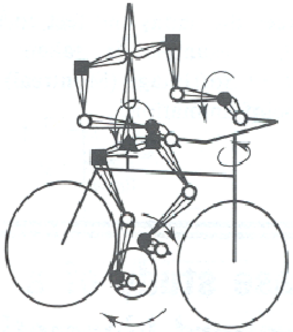
- Measures and records action performed by actor for immediate or delayed replication.
- E.g. optical: small reflective captors (markers), attached to body of real actor
- From markers position: calculate joint angles
- Body shape vs skeleton

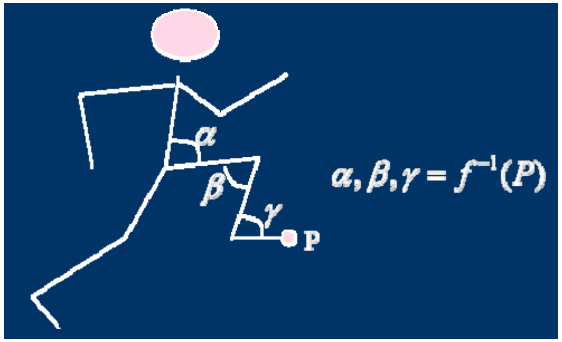
Direct and Inverse Kinematics

- **Direct Kinematics**
  - Specify joint angles
  - Compute positions of end-effectors



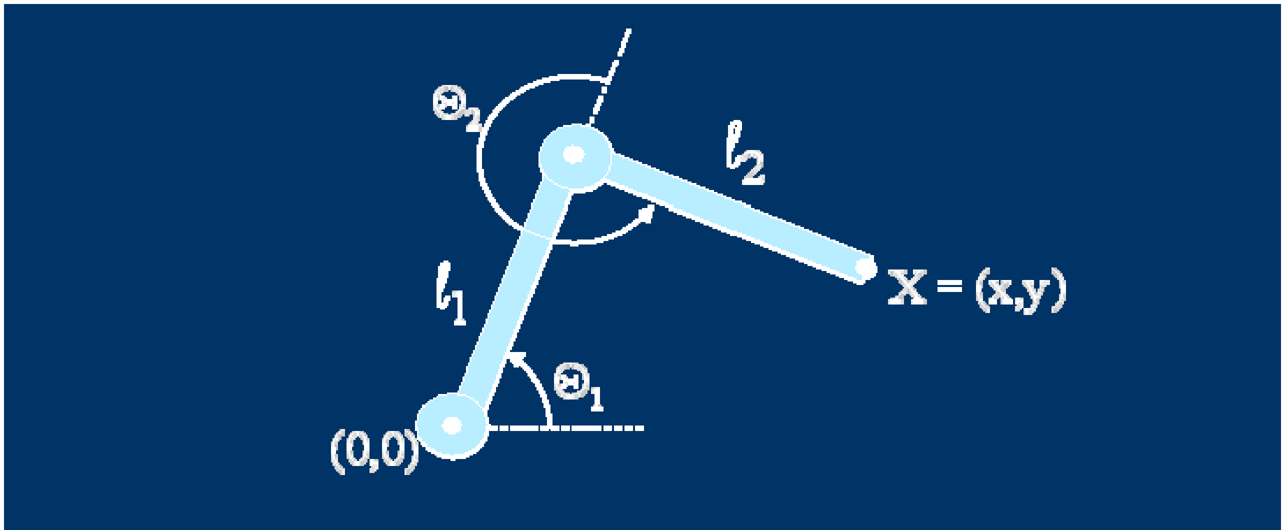
- **Inverse Kinematics**
  - “Goal-directed” motion
  - Specify goal positions of end effectors
  - Compute joint angles required to achieve goals





Direct Kinematics

- Case study: 2 links connected by rotational joints:
- Input: joint angles θ_1, θ_2 Output: position of end-effector X



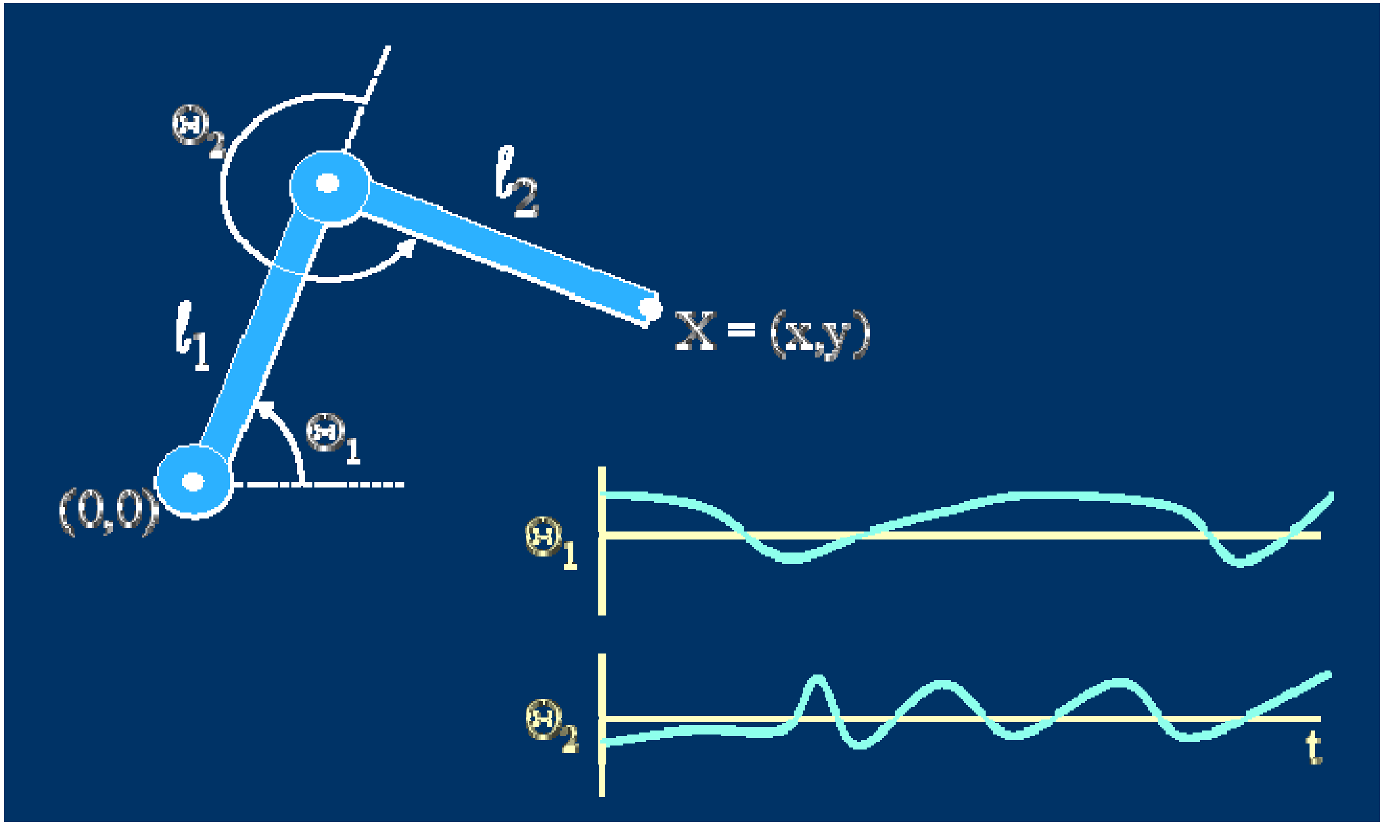
$$X = (l_1 \cos\Theta_1 + l_2 \cos(\Theta_1 + \Theta_2), l_1 \sin\Theta_1 + l_2 \sin(\Theta_1 + \Theta_2))$$

- More general case: use of Denavit-Hartenberg Matrices

$$\mathbf{A_k = R_{z\theta} T_{zd} T_{xl} R_{x\alpha}}$$

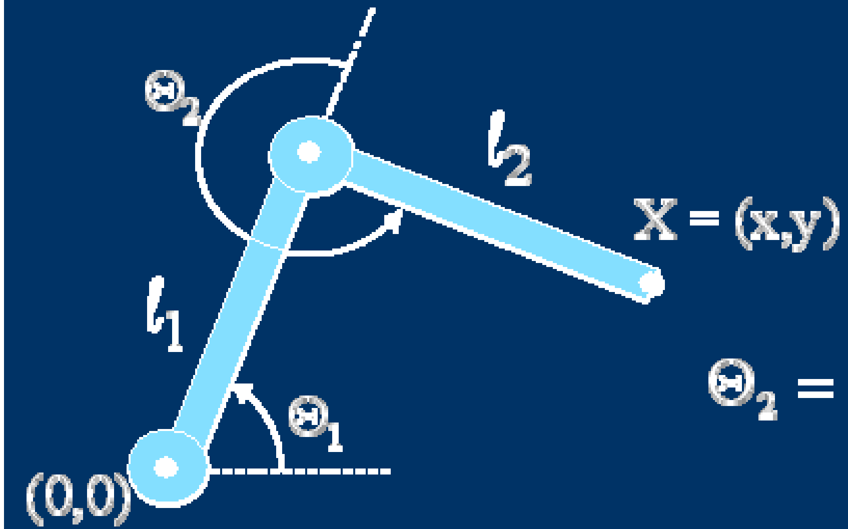
$$\mathbf{W_6 = A_1 A_2 A_3 A_4 A_5 A_6}$$

- Joint motions can be specified by spline curves
  => parametric keyframe animation



The problem of inverse kinematics

- User specifies end-effector positions: X
- Computer finds joint angles: θ_1, θ_2

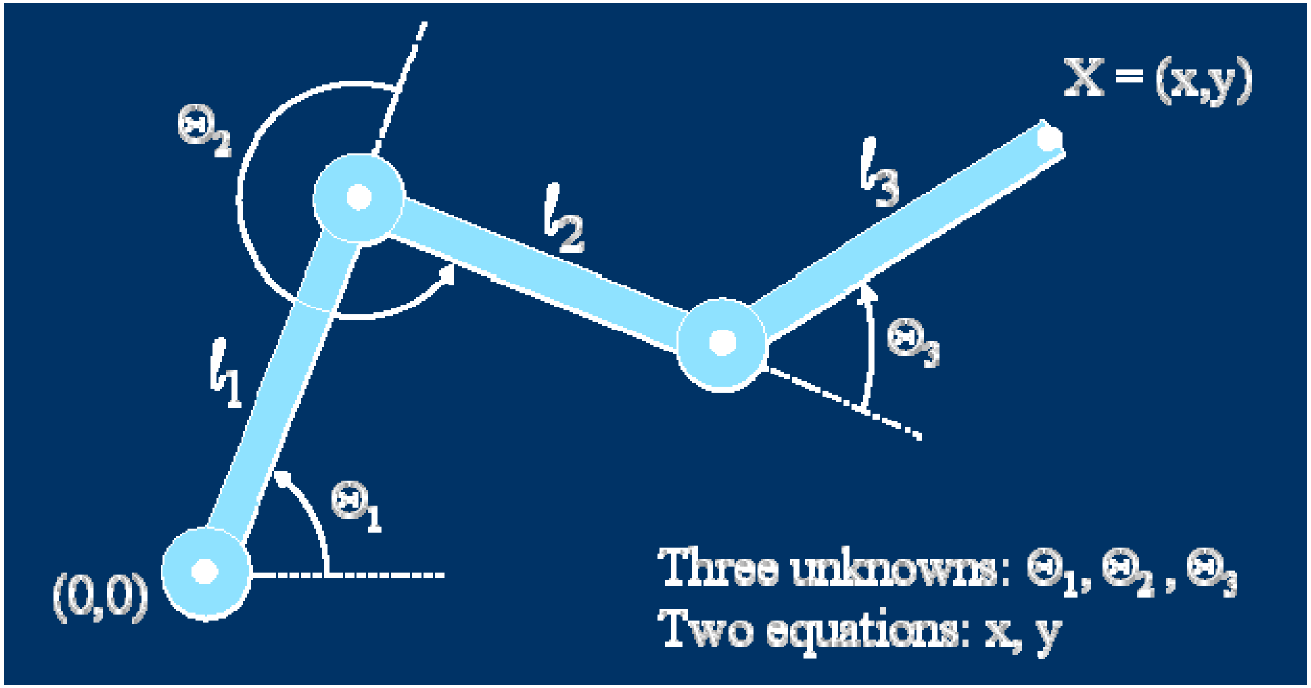


$$\Theta_2 = \cos^{-1}\left(\frac{x^2 + x^2 - l_1^{\,2} - l_2^{\,2}}{2 l_1 l_2}\right)$$

$$\Theta_1 = \frac{-(l_2 \sin(\Theta_2)x + (l_1 + l_2 \cos(\Theta_2))y}{(l_2 \sin(\Theta_2))y + (l_1 + l_2 \cos(\Theta_2))x}$$

- End-Effector Positions can be specified by Spline Curves

- **Problem for more complex structures.**
  - **System of equations usually under-defined**
  - **Multiple solutions.**
  - **Find best solution (e.g., minimize energy in motion)**
  - **Non-linear optimization**



Iterative IK Solutions

- Frequently analytic solution infeasible
- Use Jacobian
- Derivative of function output relative to each of its inputs
- If y function of 3 inputs and one output

$$y = f(x_1, x_2, x_3)$$

$$\delta y = \frac{\delta f}{\partial x_1} \cdot \delta x_1 + \frac{\delta f}{\partial x_2} \cdot \delta x_2 + \frac{\delta f}{\partial x_3} \cdot \delta x_3$$

- Relates velocities in parameter space to velocities of outputs

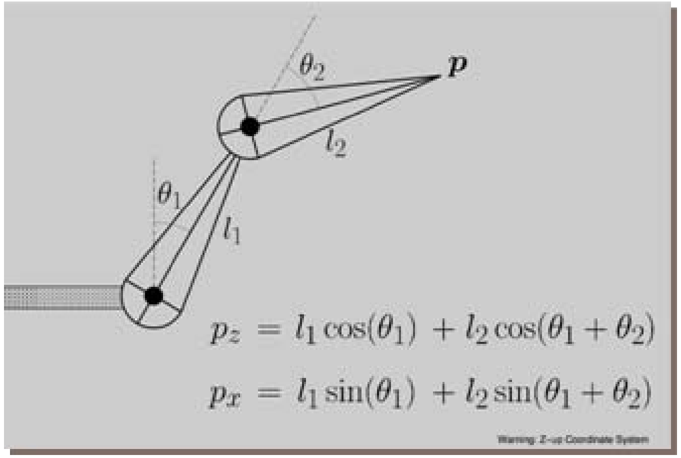
$$\dot{Y} = J(X) \cdot \dot{X}$$

- If we know $Y_{current}$ and $Y_{desired}$, then we subtract to compute $\dot{Y}$
- Invert Jacobian and solve for $\dot{X}$

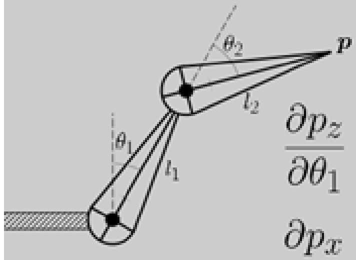
partial derivatives

Inverse Kinematics

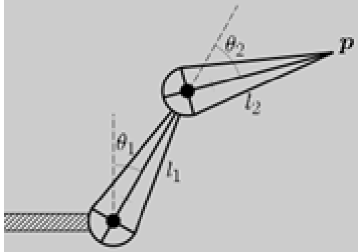
- Numerical Solution
  - Start in some initial configuration
  - Define error metric (*e.g.* goal pos - current pos)
  - Compute Jacobian of error w.r.t. inputs
  - Apply Newton's method (or other procedure)
  - Iterate...



Inverse Kinematics (two segment arm)



$$
\begin{aligned}
\frac{\partial p_z}{\partial \theta_1} &= -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) \\
\frac{\partial p_x}{\partial \theta_1} &= l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\
\frac{\partial p_z}{\partial \theta_2} &= - l_2 \sin(\theta_1 + \theta_2) \\
\frac{\partial p_x}{\partial \theta_2} &= + l_2 \cos(\theta_1 + \theta_2)
\end{aligned}
$$



Direction in Config. Space

$$
\begin{aligned}
\theta_1 &= c_1 \theta_* \\
\theta_2 &= c_2 \theta_*
\end{aligned}
$$

$$
\frac{\partial p_z}{\partial \theta_*} = c_1 \frac{\partial p_z}{\partial \theta_1} + c_2 \frac{\partial p_z}{\partial \theta_2}
$$

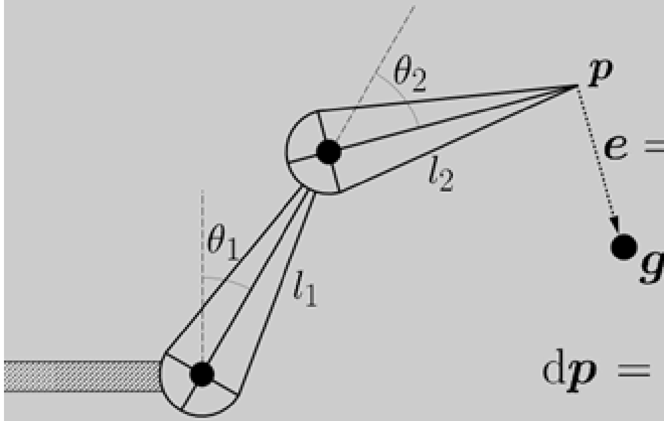
Inverse Kinematics

The Jacobian (of $\boldsymbol{p}$ w.r.t. θ)

$$J = \begin{bmatrix} \frac{\partial p_z}{\partial \theta_1} & \frac{\partial p_z}{\partial \theta_2} \\ \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} \end{bmatrix}$$

$$\frac{\partial \boldsymbol{p}}{\partial \theta_*} = J \cdot \begin{bmatrix} \frac{\partial \theta_1}{\partial \theta_*} \\ \frac{\partial \theta_2}{\partial \theta_*} \end{bmatrix} = J \cdot \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

Solving for c_1 and c_2



$$\boldsymbol{e} = \mathrm{d}\boldsymbol{p}$$

$$\mathrm{d}\boldsymbol{p} = J \cdot \boldsymbol{c}$$

$$\boldsymbol{c} = J^{-1} \cdot \mathrm{d}\boldsymbol{p}$$

Is the Jacobian invertible?

Inverse Kinematics

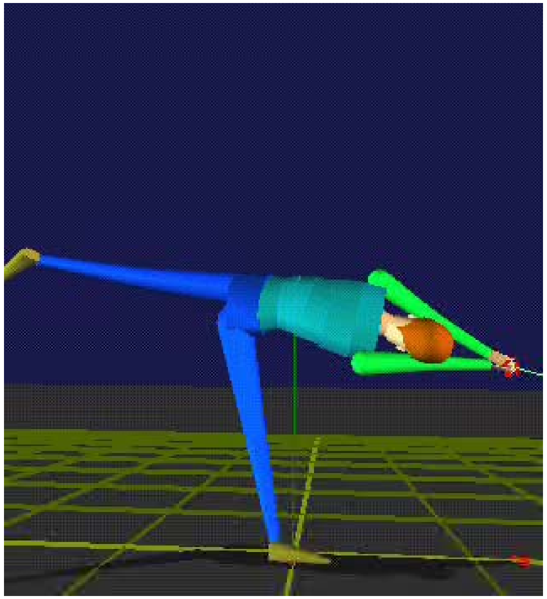
- Problems
  - Jacobian may (will!) not always be invertible
    - Use pseudo inverse (SVD)
    - Robust iterative method
  - Jacobian not constant

$$J = \begin{bmatrix} \frac{\partial p_z}{\partial \theta_1} & \frac{\partial p_z}{\partial \theta_2} \\ \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} \end{bmatrix} = J(\theta)$$

    - Nonlinear optimization, but problem (mostly) well behaved

Inverse Kinetics

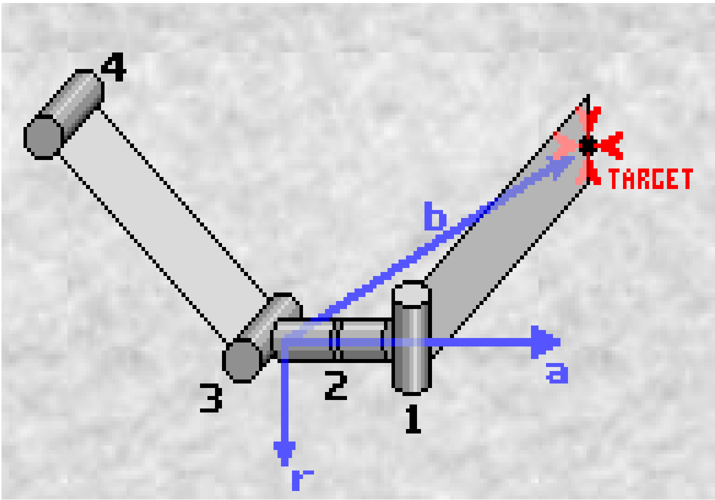
- **Better, because take into account balance**
- **With it, character cannot bend forward too much as with inverse kinematics**

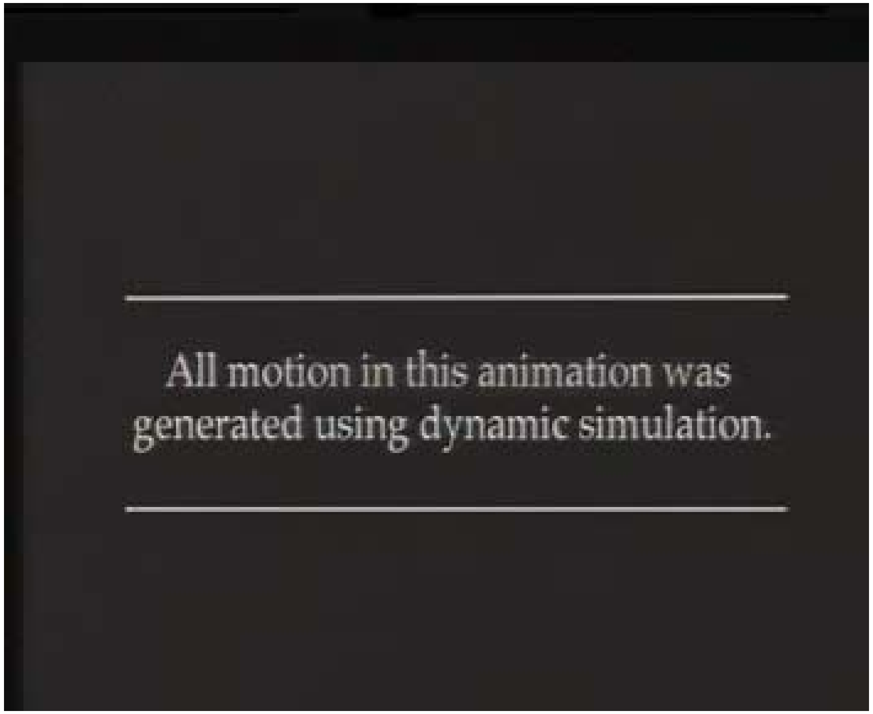


R.Boulic, R.Mas, D. Thalmann**, Complex Character Positioning Based on a Compatible Flow Model of Multiple Supports**, *IEEE Transactions in Visualization and Computer Graphics* , Vol.3, No3, 1997, pp.245-261

Dynamics

Use forces, torques, constraints and mass properties of objects

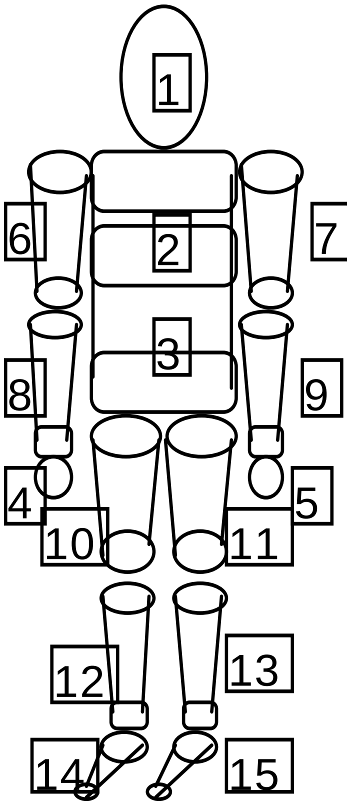




Various Dynamic Formulations

- **Newton-Euler formulation**
  1. Equations: angular / linear velocities /accelerations of each segment
  2. Equations : forces / torques exerted on successive segments.
  3. Newton equation : $F = m a$ Euler equation: $N = J \alpha + \omega \times (J.\omega)$
- **Lagrange formulation**
  - Kinematic/potential energies in terms of matrices & derivatives
  - Unfortunately, CPU time expensive: $O(n^4)$
  - Hollerbach: recursive method, Armstrong et al. almost real-time
- **D'Alembert principle of virtual works**:
  - if system in dynamic equilibrium and small movements
  - $\Rightarrow$ sum of work of forces applied & internal = work caused by change
- **Gibbs–Appel formulation**
  - Large matrices, CPU cost (O(n4))

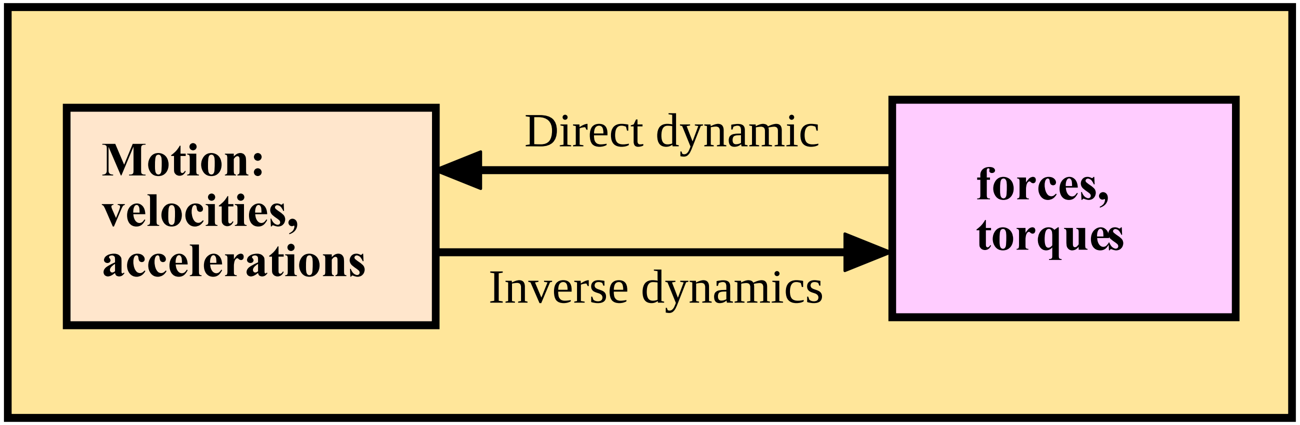
- **For dynamic simulation, volume of human body should be modelled**
- **Approximation of body by 15 solid rigid primitives; cylinders, spheres, ellipsoids and troncated cones.**
- **Mass of each volume derived from biomechanical experiments**
- **coordinate system for each solid with origin at hinge.**



Name	Type	Mass (Segment / Total)
head and neck	ellipsoid	0.081
up_torso	cylinder	0.216
down_torso	cylinder	0.281
r_hand	sphere	0.006
l_hand	sphere	0.006
r_upper_arm	frustum	0.028
l_upper_arm	frustum	0.028
r_forearm	frustum	0.016
l_forearm	frustum	0.016
r_thigh	frustum	0.100
l_thigh	frustum	0.100
r_leg	frustum	0.0465
l_leg	frustum	0.0465
r_foot	frustum	0.0145
l_foot	frustum	0.0145

Major problem in control of movement

- **Obtain value of torque produced by muscle at given joint to carry out desired movement.**
- **Use formulation of inverse dynamics based on equation of Newton-Euler to obtain values of force and torque (variables over time) for simulation in direct dynamics.**

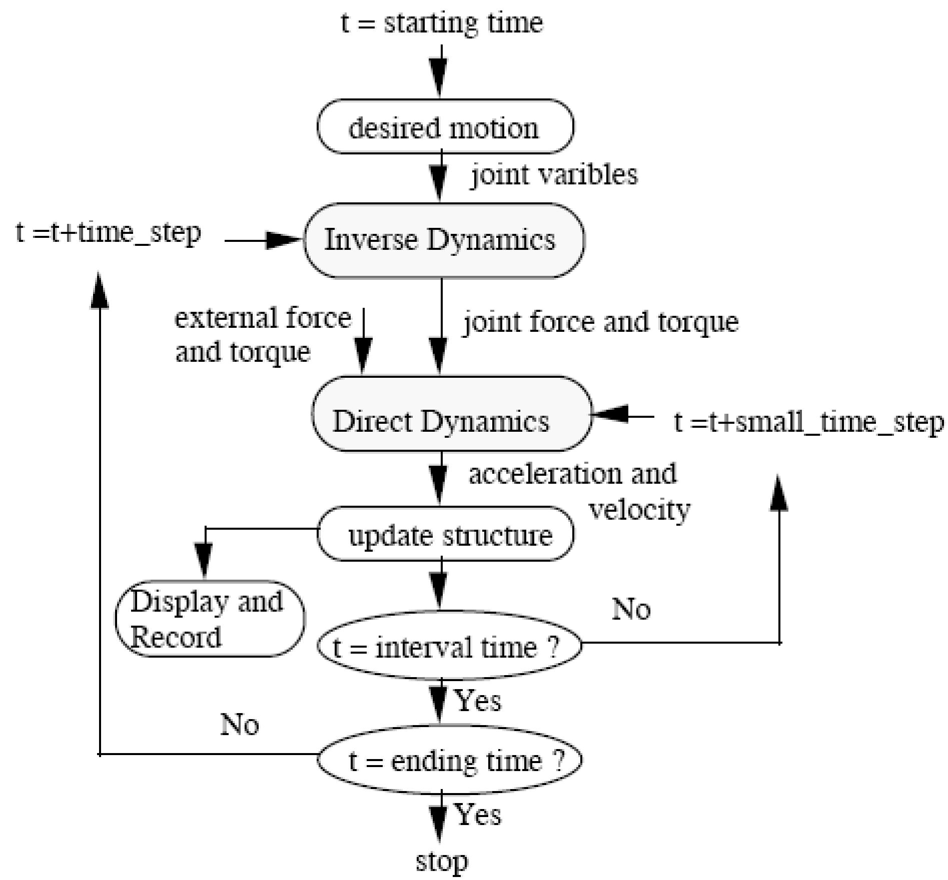


Control with inverse dynamics

- Use of inverse dynamics jointly with direct dynamics
- Expected movement: represented by joint variables

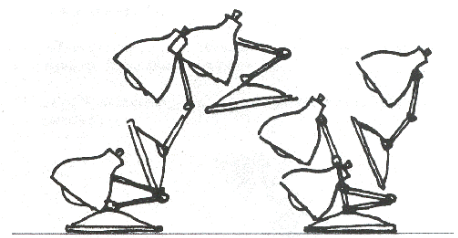
Δt: time step for inverse dynamics.

$\Delta t'$: time step for direct dynamics.



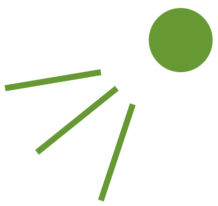
Space Time Constraints

- **Animator Specifies Constraints**
  - What the character's physical structure is
    - e.g., articulated figure
  - What the character has to do
    - e.g., jump from here to there within time t
  - What other physical structures are present
    - e.g., floor to push off and land
  - How the motion should be performed
    - e.g., minimize energy



Space Time Constraints

- **Computer Finds "Best" Physical Motion**
  - Satisfying constraints
- **Example: Particle with Jet Propulsion**
  - $\boldsymbol{x}(t)$ is position of particle at time t
  - $\boldsymbol{f}(t)$ is force of jet propulsion at time t
  - Particle's equation of motion is:



$$mx'' - f - mg = 0$$

  - Suppose we want to move from a to b within t_0 to t_1 with minimum jet fuel:

$$\text{Minimize} \int_{t_0}^{t_1} \left| f(t) \right|^2 dt \text{ subject to } x(t_0) = a \text{ and } x(t_1) = b$$

Space Time Constraints

- **Discretize Time Steps**

$$x' = \frac{x_i - x_{i-1}}{h}$$

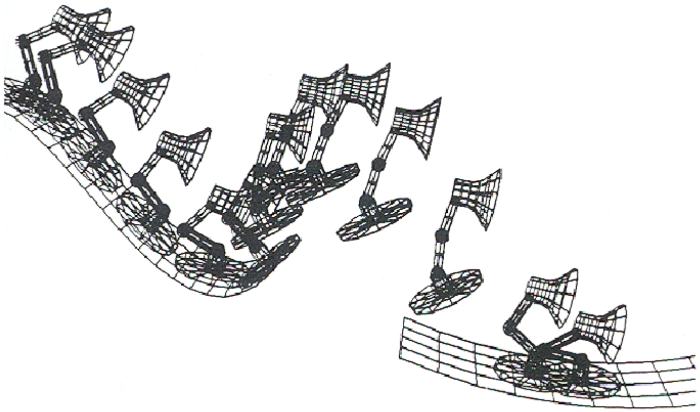
$$x'' = \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2}$$

$$m\left(x'' = \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2}\right) - f_i - mg = 0$$

$$\text{Minimize } h\sum_i |f_i|^2 \text{ subject to } x_0 = a \text{ and } x_1 = b$$

Space Time Constraints

- **Advantages**
  - Free animator from having to specify details of physically realistic motion with spline curves
  - Easy to vary motions due to new parameters and/or new constraints

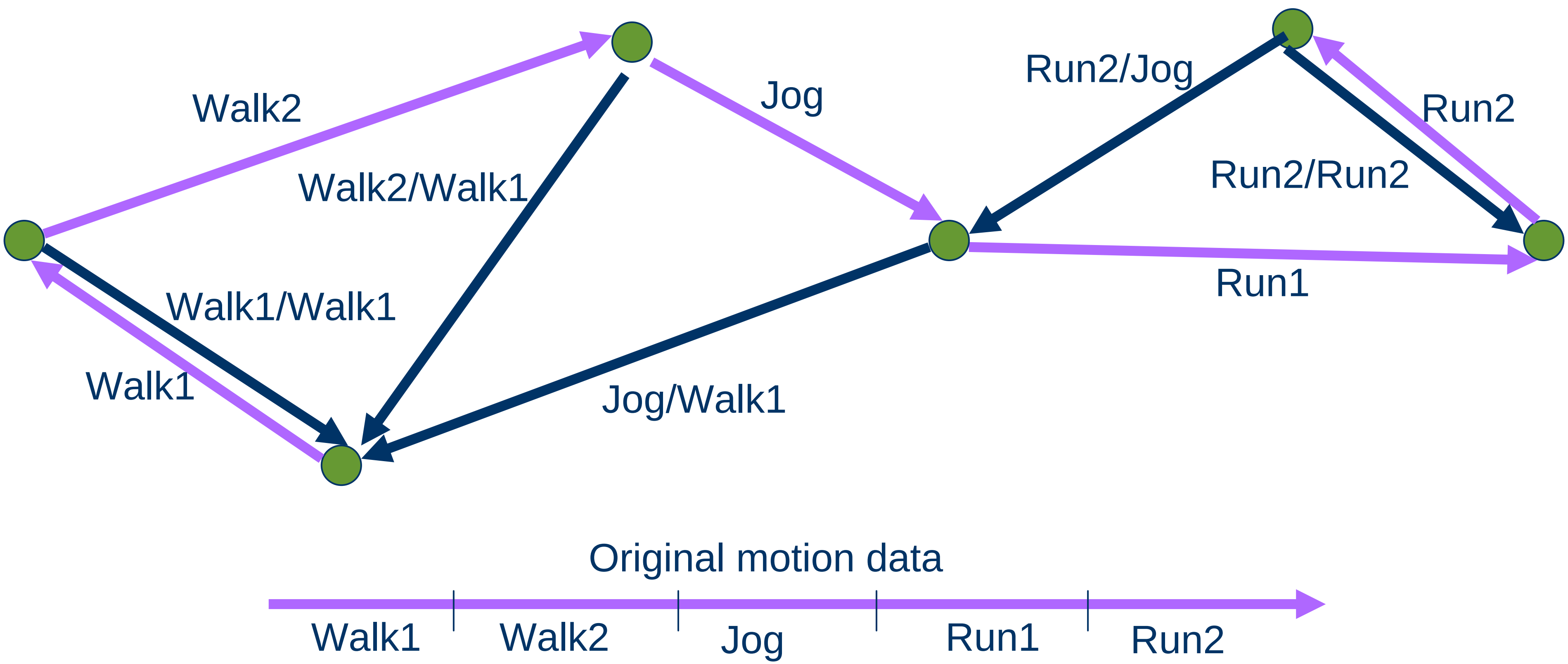


- **Challenges**
  - Specifying constraints and objective functions
  - Avoiding local minima during optimization

Motion graphs

- Roadmap of motion data for human
  - Edges: clips of motion, some from original motion capture, some generated as transitions
  - Vertices: 2 sets of motion clips where motions from one set can flow seamlessly into motions from the other
- Building motion graphs
  - Identify transition candidates
  - Select transition points
  - Eliminate problematic edges

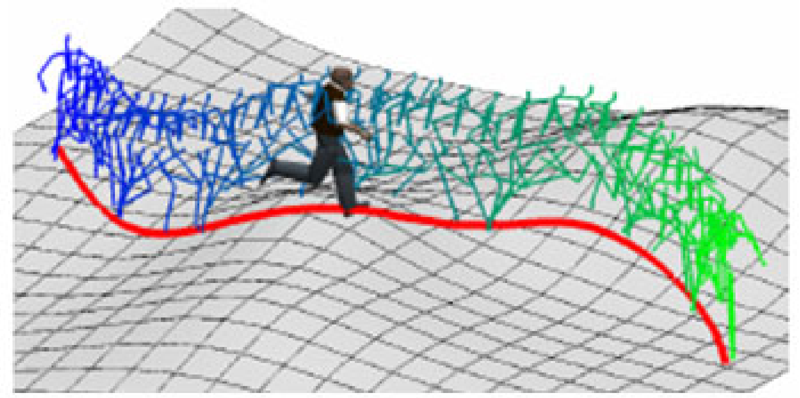
A simple motion graph



Identify transition candidates

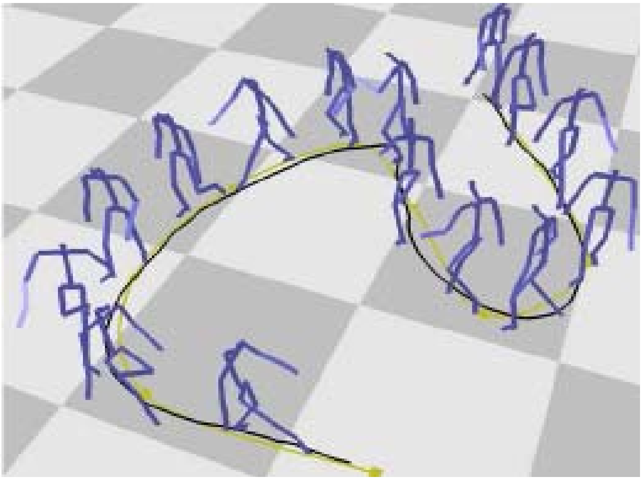
- For each frame A, calculate distance to each other frame B by basically measuring volume displacement
- Use weighted point cloud formed over window of k frames ahead of A and behind B, ideally from body mesh
- Calculate minimal weighted sum of squared distances between corresponding points, given that rigid 2D transformation may be applied to second point cloud

$$\min_{\theta, x_0, z_0} \sum_i w_i \| \mathbf{p_i} - \mathbf{T}_{\theta, \mathbf{x}_0, \mathbf{z}_0} \mathbf{p}'_\mathbf{i} \|^2$$



Select transition points

- Previous step gave us all local minima of distance function for each pair of points
- Now define threshold and cut transition candidates with errors above it
- May be done with or without intervention
- Threshold level depends on type of motion – eg. walking vs. ballet



Create transitions

- For each pair of frames A_i and B_j which fell under distance error threshold, blend A_i-A_{i+k-1} with B_j-B_{j-k+1}
  - Align frames with appropriate rigid 2D transformation
  - Use linear interpolation to blend root positions

$$R_p = \alpha(p) R_{\mathscr{A}_{i+p}} + [1 - \alpha(p)] R_{\mathscr{B}_{j-k+1+p}}$$

  - Use spherical linear interpolation to blend joint rotations

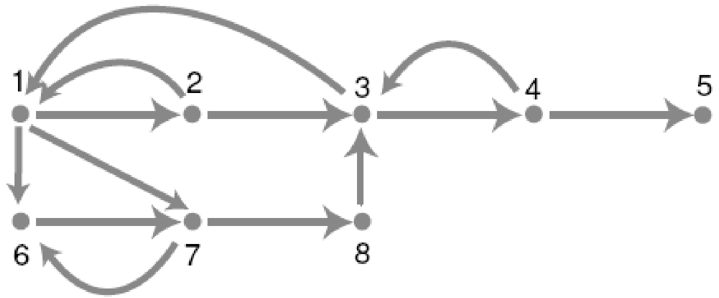
$$q_p^i = slerp(q^i_{\mathscr{A}_{i+p}}, q^i_{\mathscr{B}_{j-k+1+p}}, \alpha(p))$$

  - Blend function:

$$\alpha(p) = 2\left(\frac{p+1}{k}\right)^3 - 3\left(\frac{p+1}{k}\right)^2 + 1, \; -1 < p < k$$

Eliminate problematic edges

- We want to get rid of:
  - Dead ends – not part of a cycle
  - Sinks – part of one or more cycles but only able to reach small fraction of nodes
  - Logical discontinuities – eg. boxing motion forced to transition into ballet motion
- Goal: to be able to generate arbitrarily long streams of motion of same type



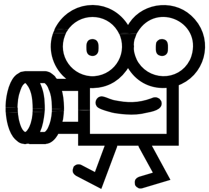
Using motion graphs

- We have: database of motion segments and mappings between them
- We want: motion streams conform to user specifications
- Search problem, where user specifies non-negative scalar error function g(w,e) and halting condition
- Total error of path w:

$$f(w) = f([e_1, \ldots, e_n]) = \sum_{i=1}^{n} g([e_1, \ldots, e_{i-1}], e_i)$$

Searching

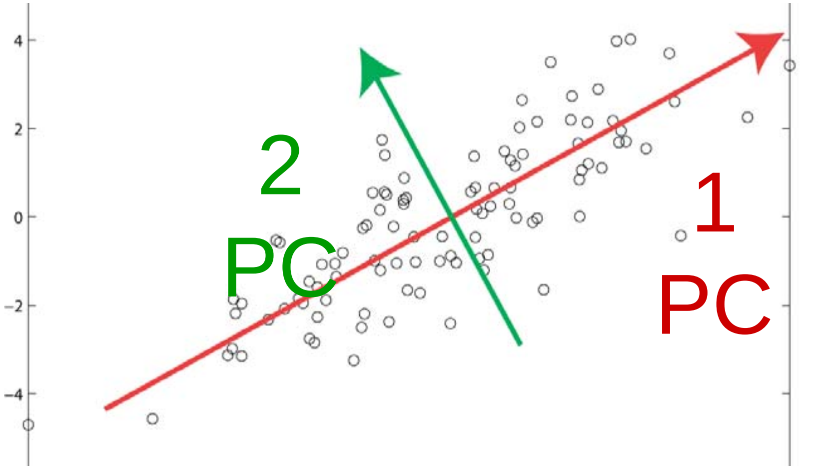
- Goal: find complete graph walk that minimizes f(w)
- Use branch and bound – keep track of best complete graph walk w_{opt} and cut current branch when error exceeds $f(w_{opt})$
- Use greedy ordering heuristic – for set of unexplored child nodes, select one that minimizes g(w,c)
- Even with branch & bound and ordering heuristic, search still exponential
- Trade some optimality for speed by searching incrementally



Motion Modeling: Creation of PCA space

- **PCA technique transforms set of variables into smaller set of its linear combination, with most of variance of original dataset.**

$$\theta \approx \theta_0 + \sum_{i=1}^{m} \alpha_i e_i = \theta_0 + \alpha \mathbf{E}^T$$



Motion Modeling: Time Warping

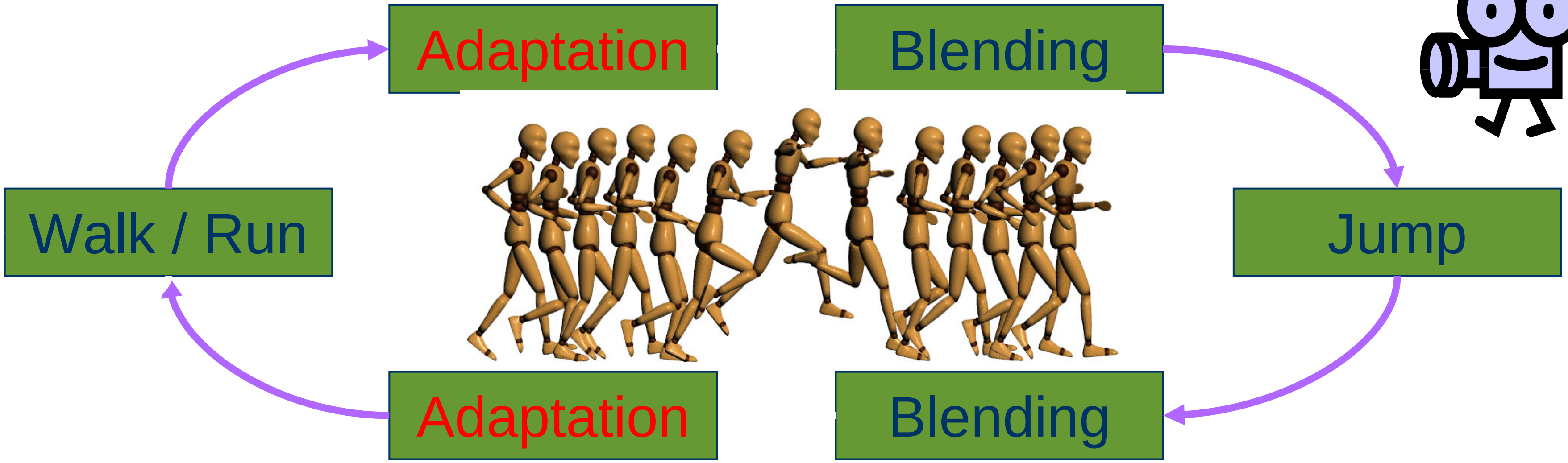
- Space ➡ leg length
- Time ➡ original duration
  - Frequency function $F(param)$
  - Phase φ

$$\varphi_{i+1} = \varphi_i + \Delta\varphi = \varphi_i + \frac{\Delta t}{MotionUnit\ Duration}$$

$$= \varphi_i + \Delta t \cdot F(param)$$

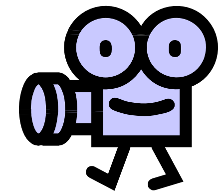
Adaptive Motion Control

- Automatic on-line motion transition
  - Locomotion / Jump
  - Time and duration → identical footplant
  - Dynamic coherence → type, run-up speed

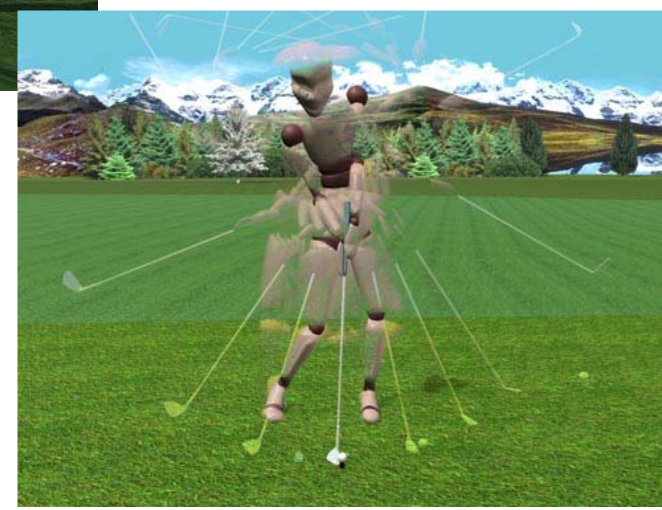


P. Glardon, R. Boulic and D. Thalmann, Robust on-line adaptive footplant detection and enforcement for locomotion, *Visual Computer*, Vol. 22, Nr. 3, pp. 194-209, 2006.

Golf



- Combining PCAs, warping, and Prioritorized Inverse Kinematics



S. Carvalho, R. Boulic, D. Thalmann, Interactive Low-Dimensional Human Motion Synthesis by Combining Motion Models and PIK, *Computer Animation & Virtual Worlds*, Vol. 18, 2007.

Conclusion

- Motion Control still strongly relies on Motion Capture
- Weak situation compared to shape modeling or rendering
- Can we get rid of Motion Capture ?