

Character Animation

Skeletal Animation

Skeleton structure

Characteristics

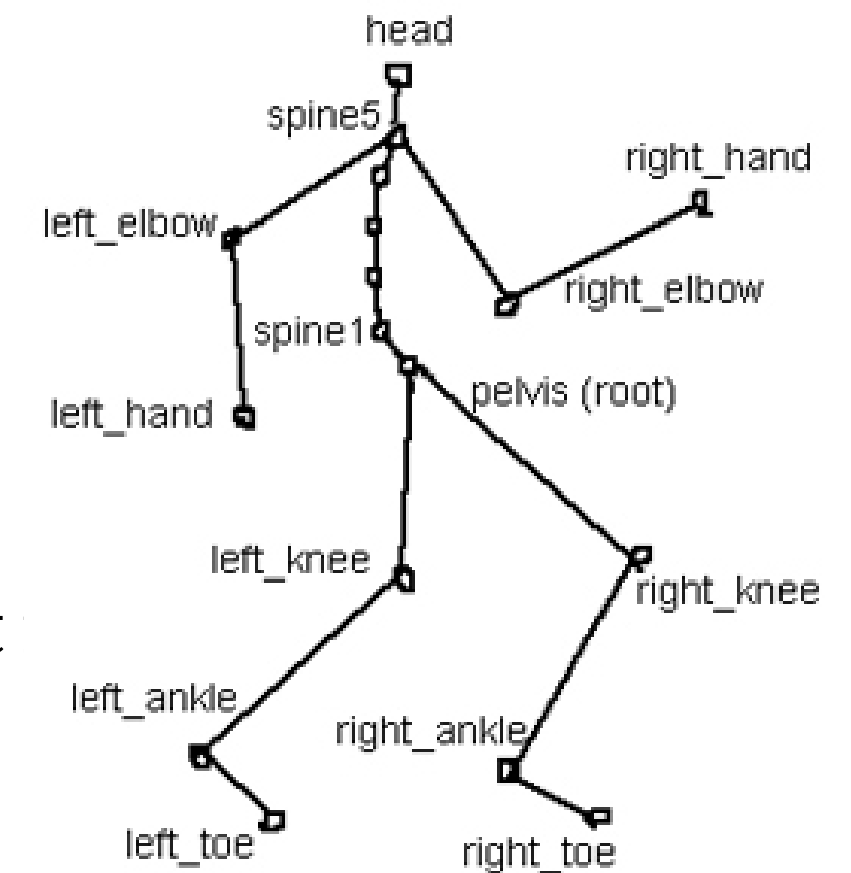
- Hierarchical representation

All children follow the transformation of the parent

Need to define a root: usually at the hips/pelvis

- Convenient to express relative transformation with respect to the parent

ex. Ankle is at 20° w/r knee



Converting local to global frames/joint coordinates

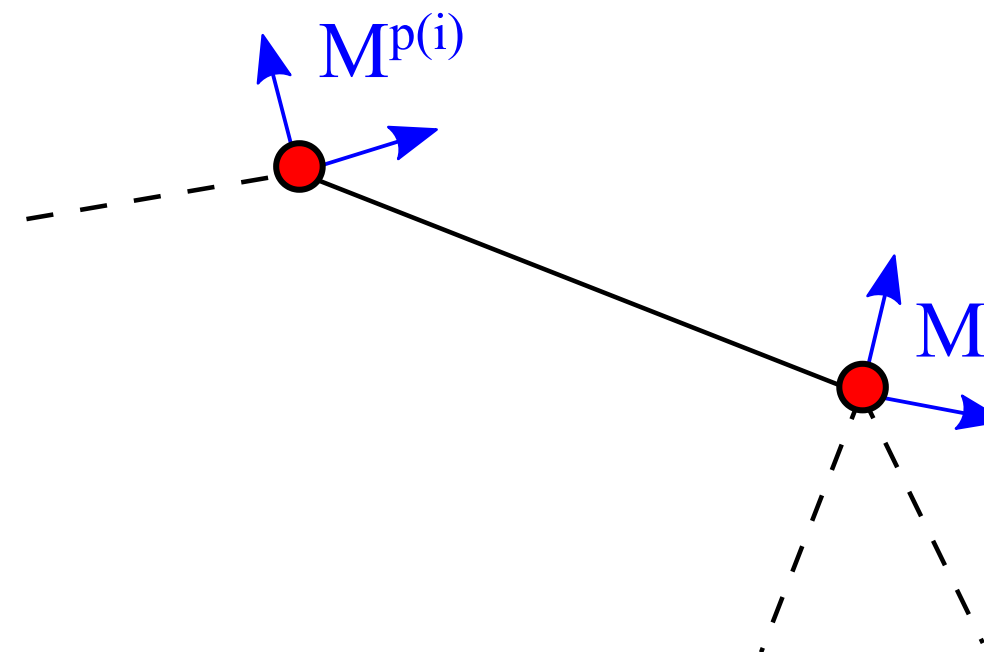
- With 4x4 matrices M

$$M_{global}^i = M_{global}^{p(i)} M_{local}^i$$

- With translation t , rotation R

$$R_{global}^i = R_{global}^{p(i)} R_{local}^i$$

$$t_{global}^i = t_{global}^{p(i)} + R_{global}^{p(i)} t_{local}^i$$



Encoding hierarchical skeleton

- Simplest encoding based on index within vector

Geometry=[M0, M1, M2, M3, M4, M5]

Parent = [-1,0,1,1,0,4]

- Convert local coordinates to global coordinates

local (Geometry) <- std::vector of rotation (r), translation (p)

global (Geometry) <- std::vector of rotation (r), translation (p)

```
global[0] = local[0];
```

```
for(size_t k=1; k<N; ++k)
```

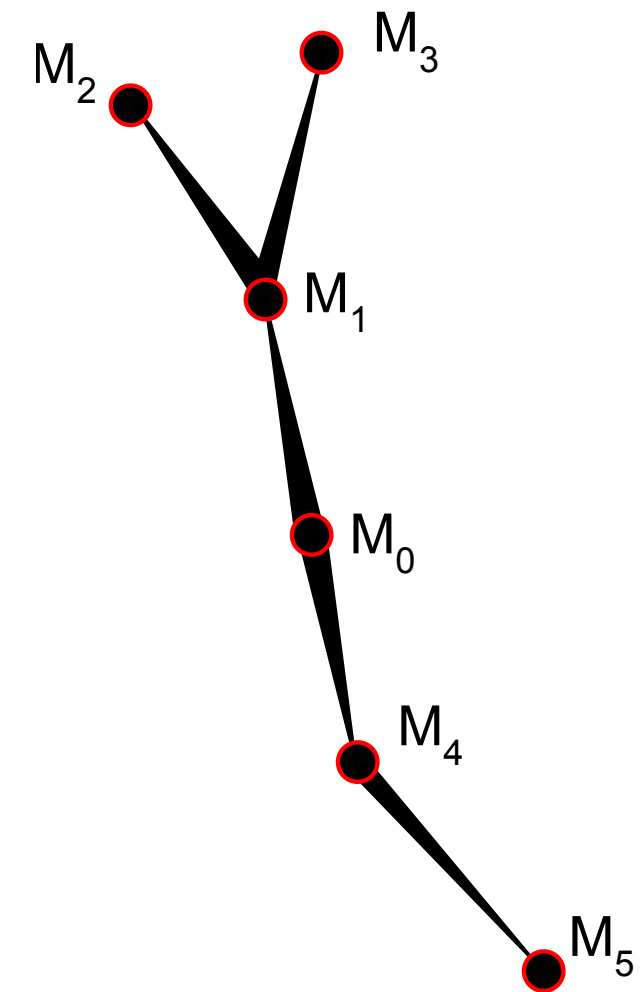
```
{
```

```
    int parent = Parent[k];
```

```
    global[k].r = global[parent].r * local[k].r;
```

```
    global[k].p = global[parent].r * local[k].p + global[parent].p;
```

```
}
```

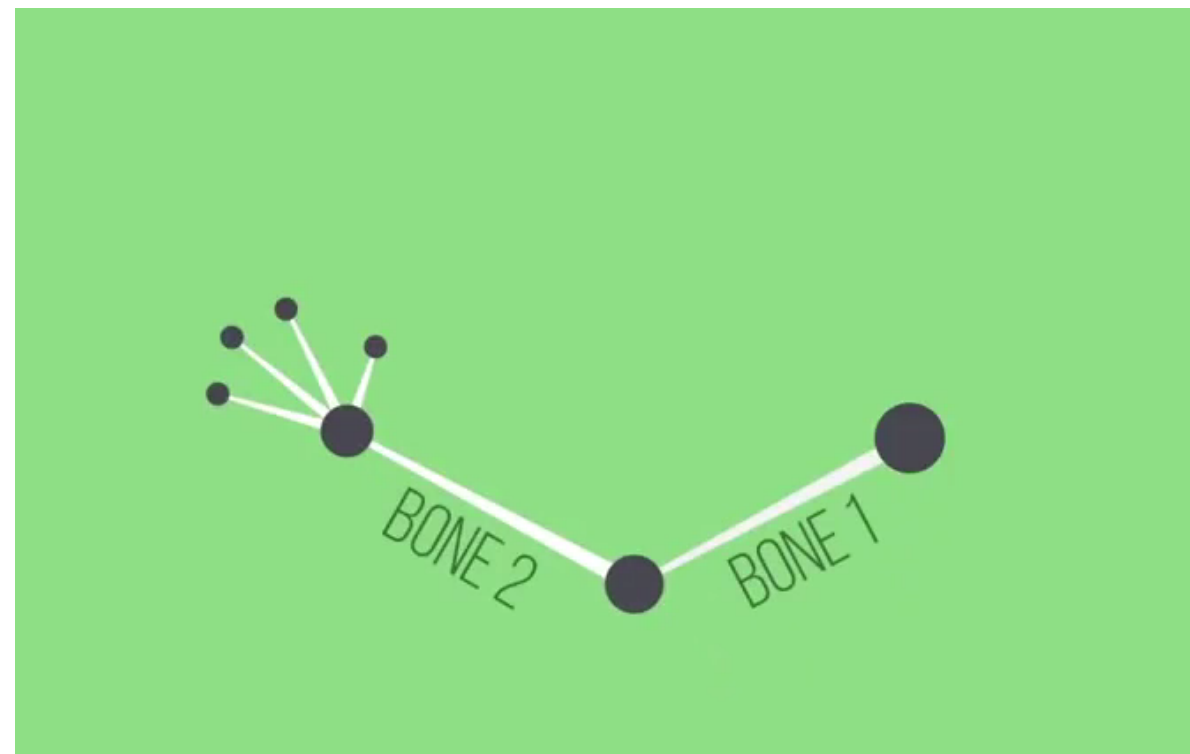
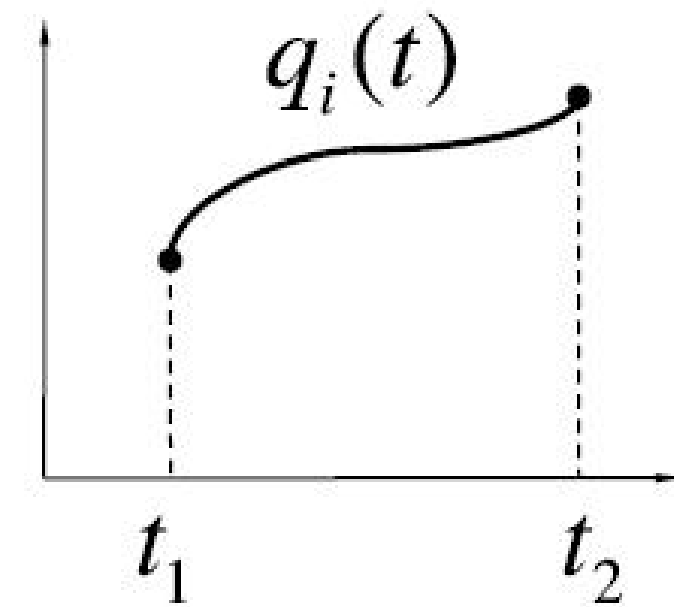
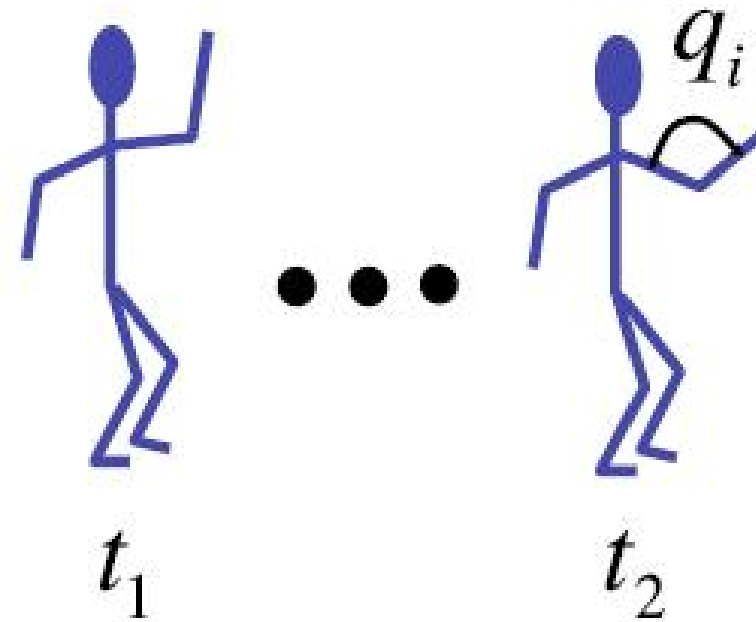


Forward kinematics

FK - Forward Kinematics

- Each joint angle is set manually
- Adapted to set orientation of specific parts
- Interpolate rotations during animation

(+) Generates curved trajectory naturally



MiloScerny Animation

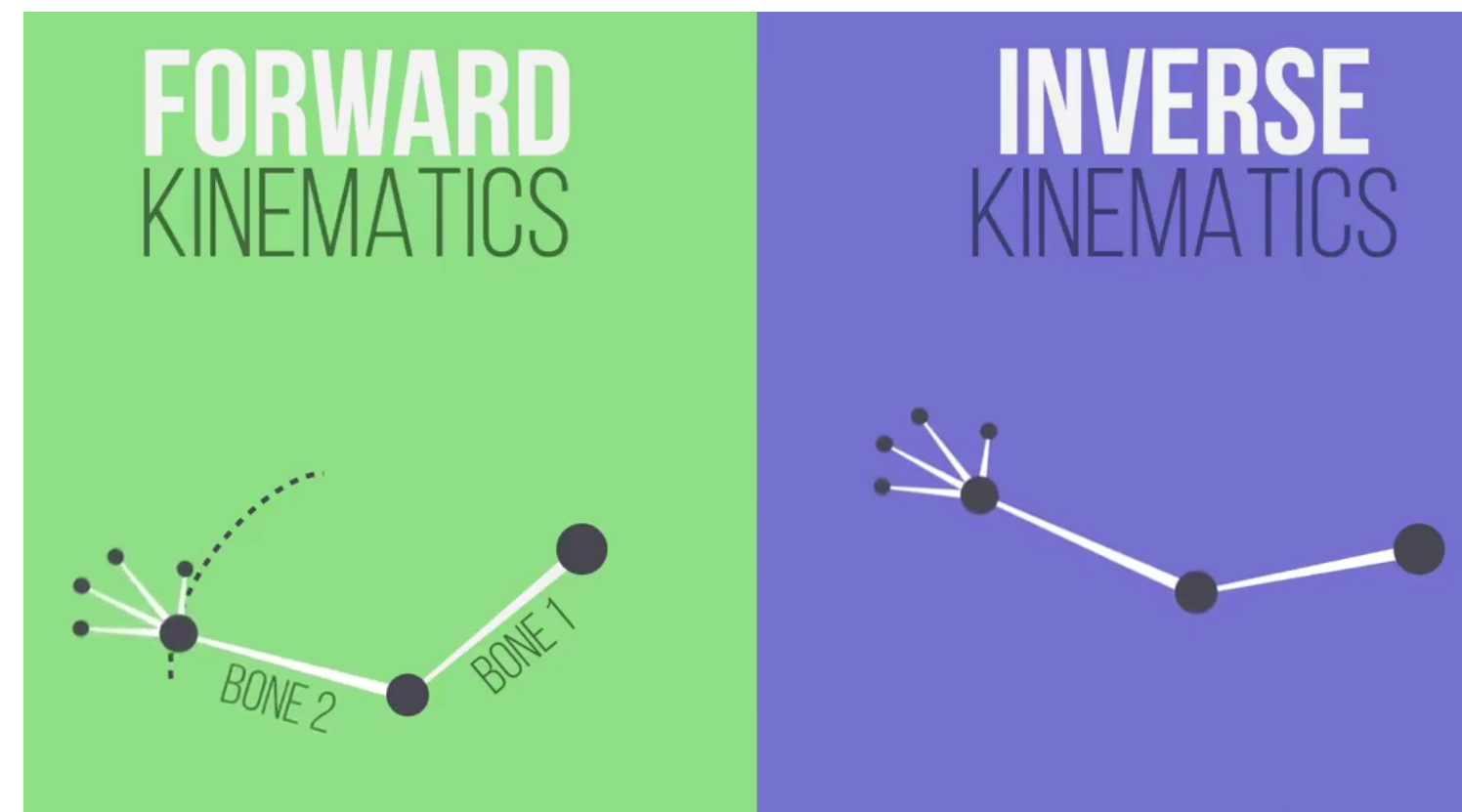
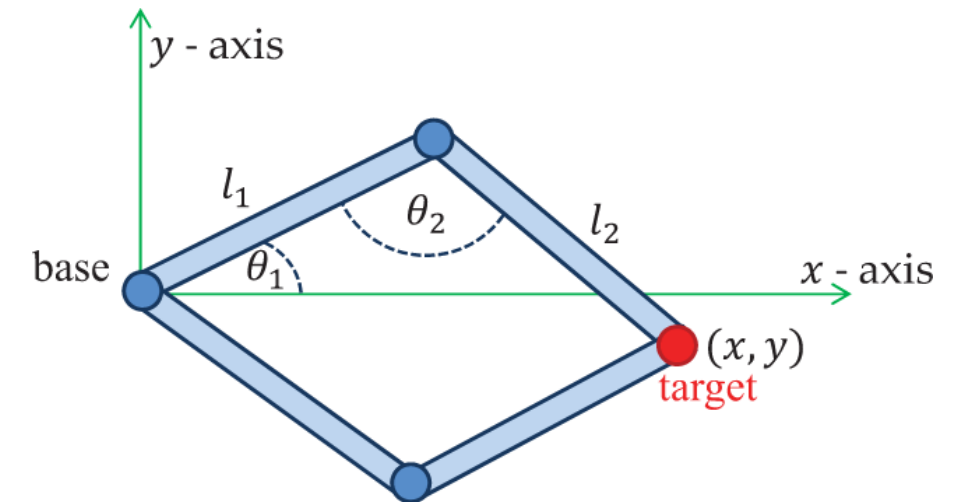
Inverse Kinematics

IK: Inverse Kinematics

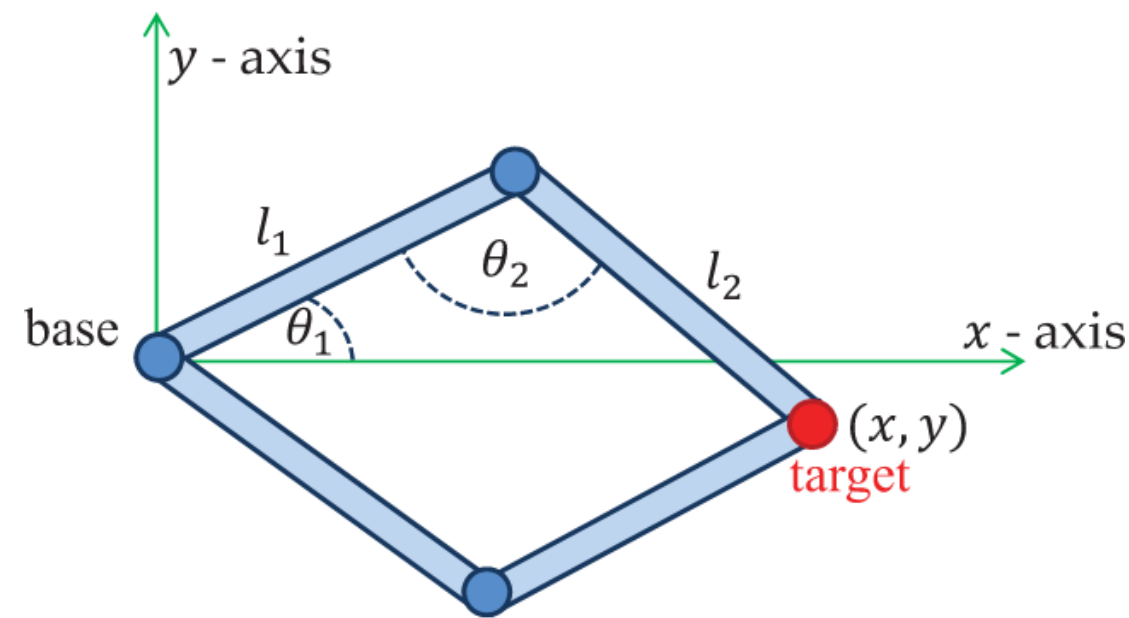
- Describe position (/and orientation) of *end-effectors* (contact, walking, etc)
- Compute joint angles reaching this position

$$p_k = p_0 + \sum_{i=0}^{k-1} l_i R_i u_i$$

R_i can be expressed with various rotation parameters (Matrices, Euler angles, axis/angle, quaternions, etc)



IK Example with two bones



[*Inverse Kinematics Techniques in Computer Graphics: A Survey.* A. Aristidou et al. STAR EG. 2017.]

In general the general case $p_k = f(\theta_i)$

- Look for $\theta_i = f^{-1}(p_k)$
- f is a non linear function
- There may exists multiple solutions (or none)
- Solutions may exhibits discontinuities
- Closed form solution are not available

Two solutions defined by

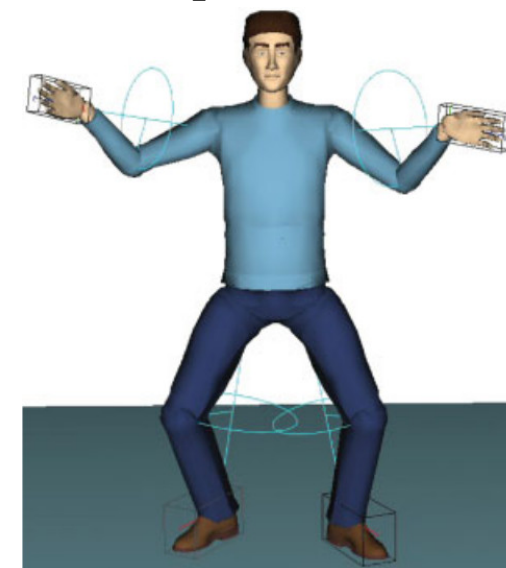
$$\cos(\theta_1) = \frac{l_1^2 + x^2 + y^2 - l_2^2}{2l_1 \sqrt{x^2 + y^2}}$$

$$\cos(\theta_2) = \frac{l_1^2 + l_2^2 - (x^2 + y^2)}{2l_1 l_2}$$

Some attempts for explicit solutions in specific cases

[*Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs.* D. Tolani et al. *Graphical Models*, 2000.]

[*Analytical inverse kinematics with bodyposture control.* M. Kallmann. *Comp. Anim. & Virt. Worlds*, 2008]



IK: Numerical methods

Numerical inversion of $p = f(\theta)$, $\theta = (\theta_0, \dots, \theta_{N-1})$.

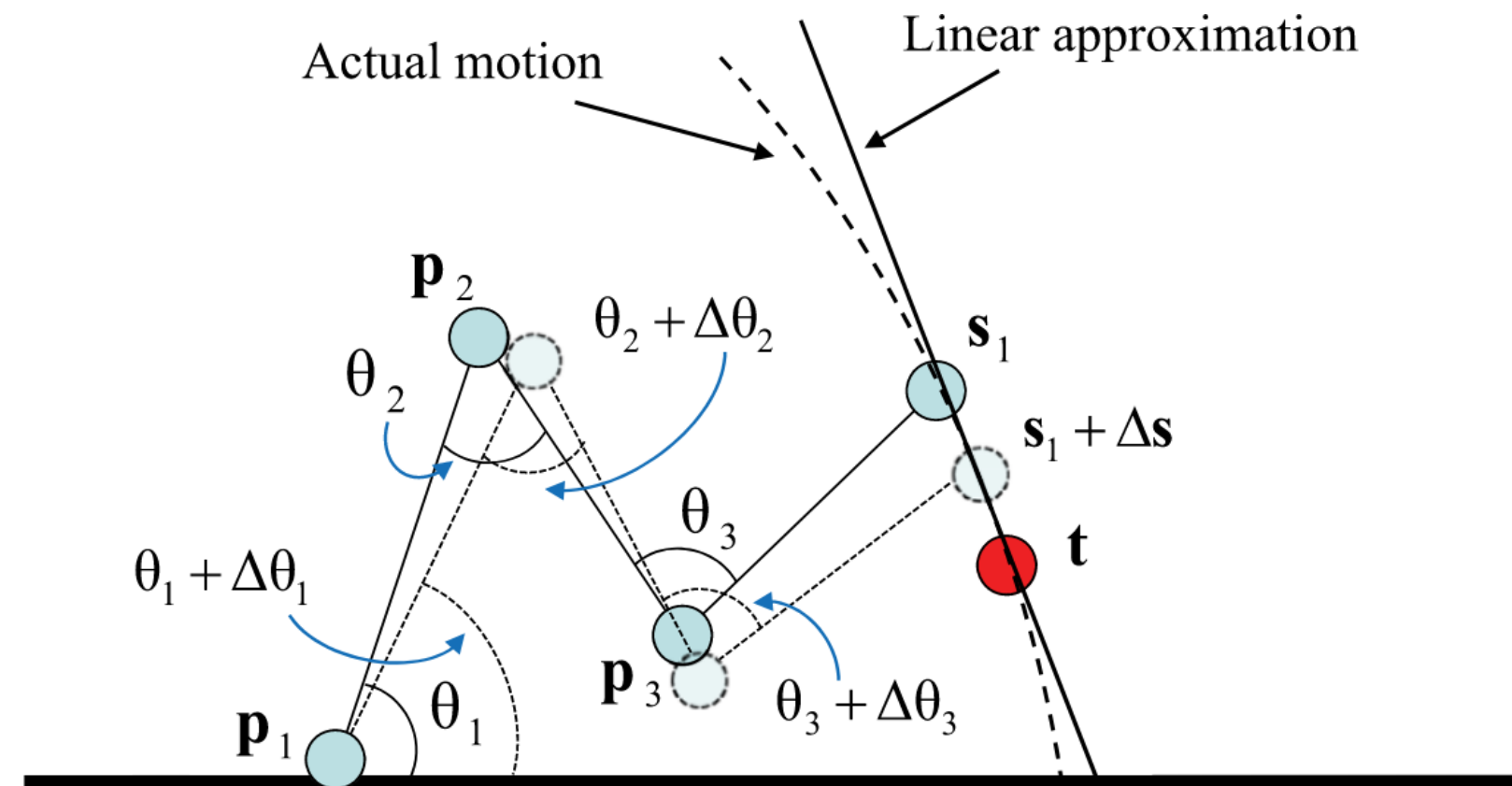
Consider small step size $p \rightarrow p + \Delta p$

$$\Delta p \simeq \underbrace{\left(\frac{\partial f}{\partial \theta} \right)}_J \Delta \theta$$

J - Jacobian matrix.

→ Not square ($3 \times N$), not invertible.

Unknown > # constraints



IK: Numerical methods

Several possible approaches to solve $\mathbf{J} \Delta\theta = \Delta p$

- Pseudo Inverse

$$\Delta\theta = \mathbf{J}^+ \Delta p, \text{ with } \mathbf{J}\mathbf{J}^+ = \mathbf{I}$$

$$\mathbf{J}^+ = \mathbf{J}^T (\mathbf{J} \mathbf{J}^T)^{-1}$$

- Can also be computed using SVD: $\mathbf{J}^+ = \mathbf{V} \Sigma^+ \mathbf{U}^T$

$$\Sigma_{ii} = \sigma_i, \Sigma_{ii}^+ = 1/\sigma_i \text{ if } \sigma_i \neq 0, 0 \text{ otherwise.}$$

- Adding damping to compensate for singularities

$$\Delta\theta = \mathbf{J}^T (\mathbf{J} \mathbf{J}^T + \lambda^2 \mathbf{I})^{-1} \Delta p$$

- Using Newton's methods

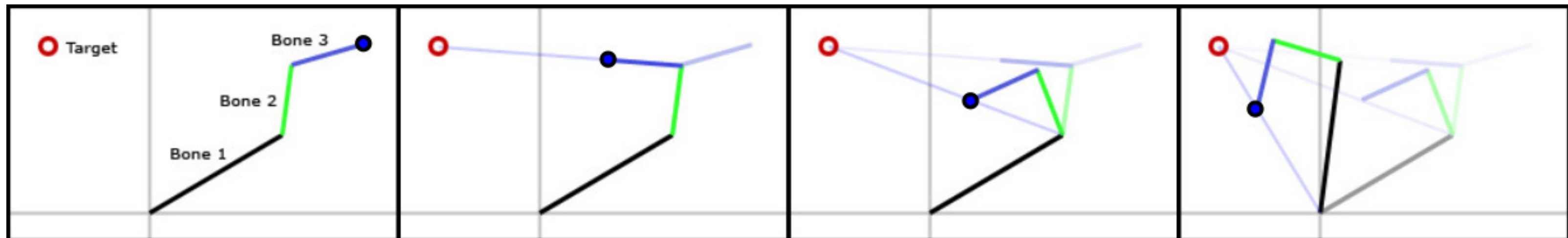
[*Inverse Kinematics Techniques in Computer Graphics: A Survey*. A. Aristidou. STAR EG 2017]

IK: Heuristic approach

Cyclic Coordinates Descent (CCD)

- Iteratively rotates joint $j^N \rightarrow j^{i-1} \rightarrow \dots \rightarrow j^1$ for the extremity (end effector) to be as close as possible from the target.
 - = *End-effector aligned with the segment (joint, target)*
- Restart until convergence

[Making Kine More Flexible. Jeff Lander. Game Dev, 1998.]



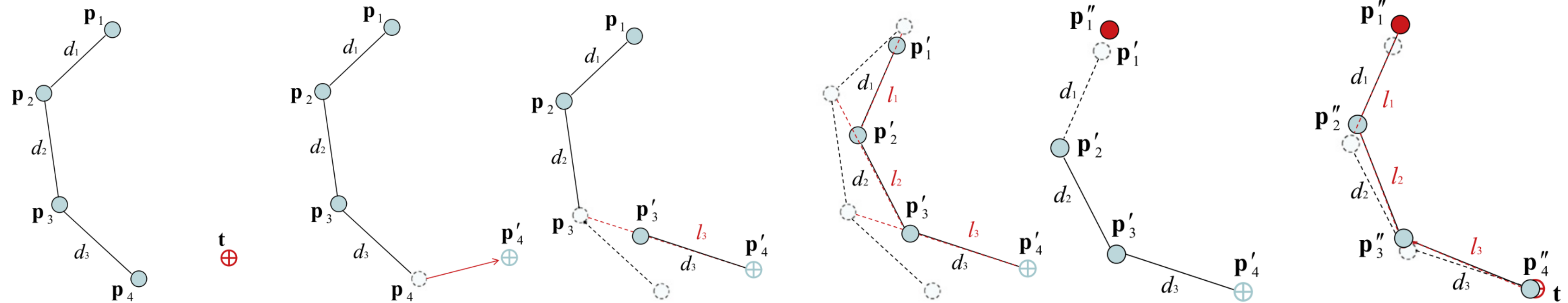
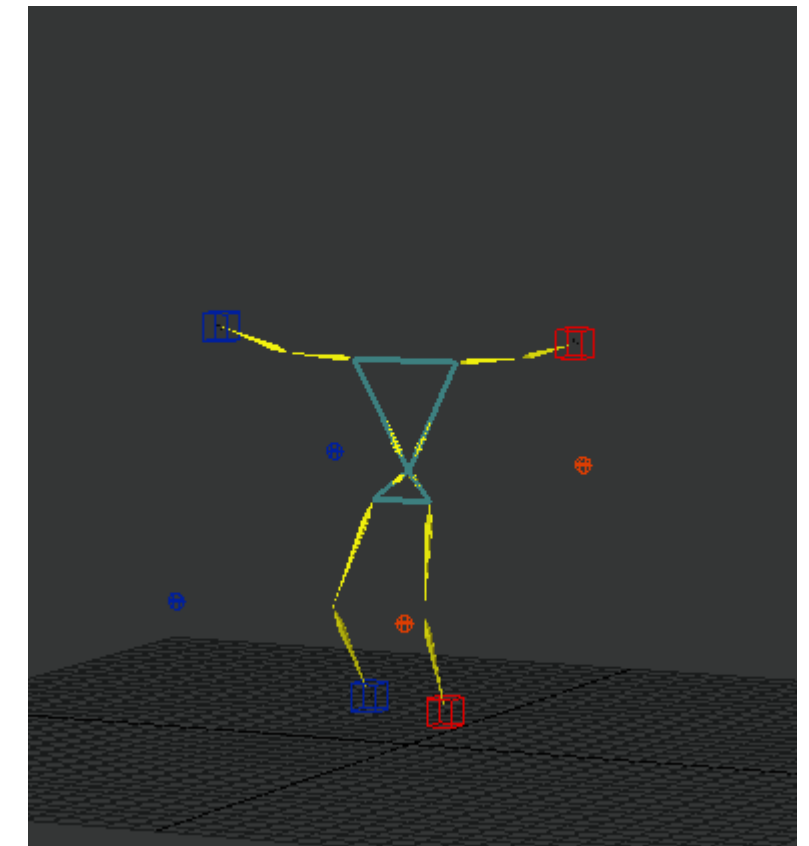
IK: Heuristic approach

Fabrik

Iterate between

- Forward direction: Match the end-effector target
propagate changes toward previous position to match bones' length.
- Backward direction: Match the starting position
propagate changes toward following positions to match bones' length.

[A fast iterative solver for the Inverse Kinematics problem. A. Aristidou. Graphical Models 2011.]



Inverse Kinematics

Example

Synthesizing and controlling skeleton animation

Blending skeleton animation

Pre-store several looping animation

Blend between animation for transition

`three.js` - Skeletal Animation Blending (model from realitymeltdown.com)

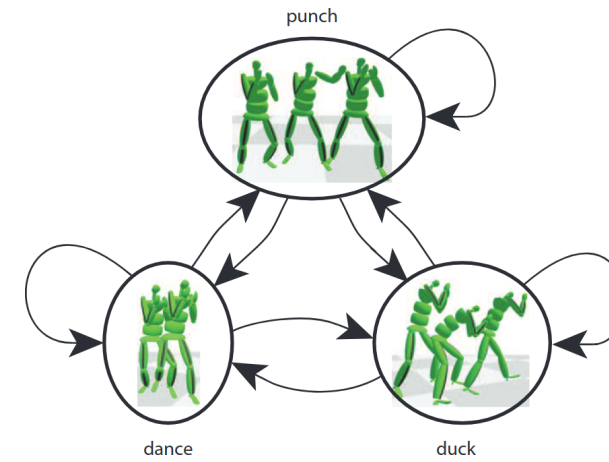
camera orbit/zoom/pan with left/middle/right mouse button

Note: crossfades are possible with blend weights being set to (1,0,0), (0,1,0) or (0,0,1)

Your graphics card does not seem to support [WebGL](#).
Find out how to get it [here](#).

Motion graphs

Also called Move Trees (highly used in video games)



- Pre-stores multiple precomputed animation

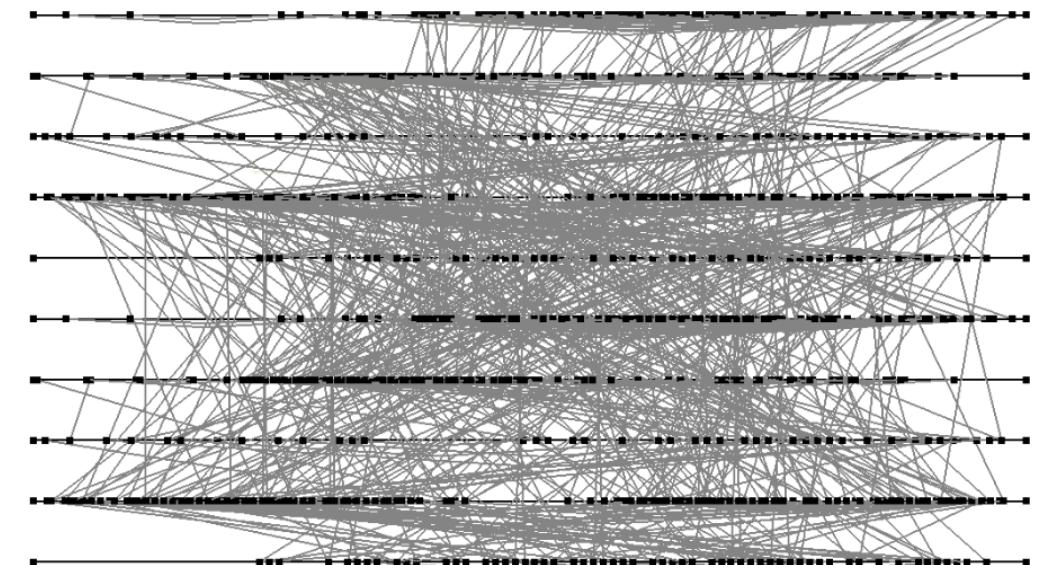
Manually design, motion capture, etc

- Find optimal transitions between different motions
- [Mizuguchi et al., Data driven motion transitions for interactive games, EG short paper, 2001]
- [Kovar et al., Motion Graphs, ACM SIGGRAPH 2002]
- [Heck and Gleicher, Parametric Motion Graphs, ACM SIGGRAPH 2007]

- More recent: Use of "Motion Matching"

Predict dynamically best possible animation frame
based on current and desired future state.

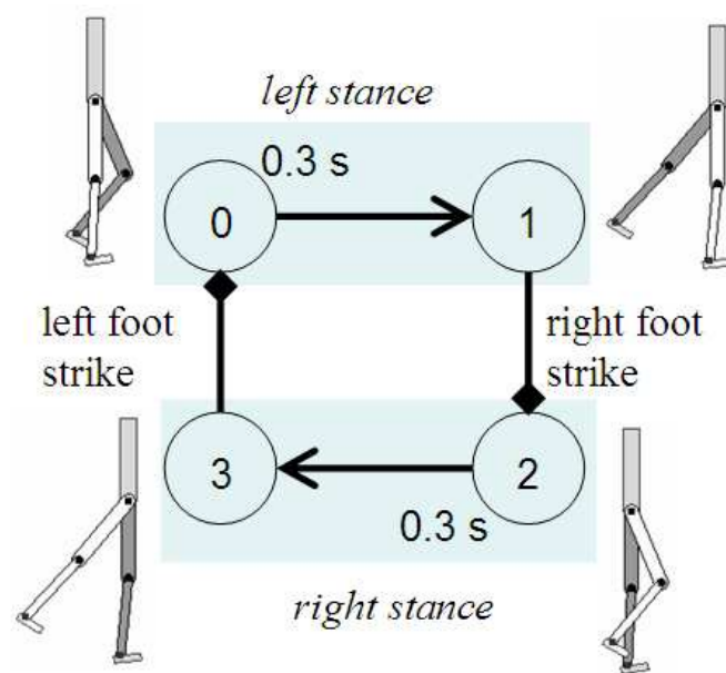
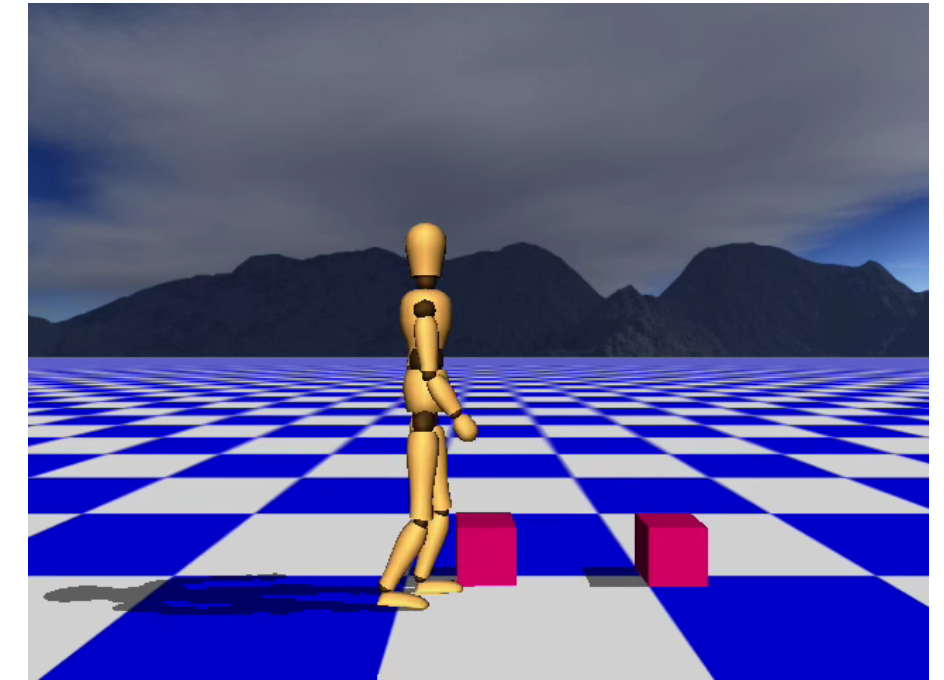
[Simon Clavet, Motion Matching and The Road to Next-Gen Animation, GDC 2016]



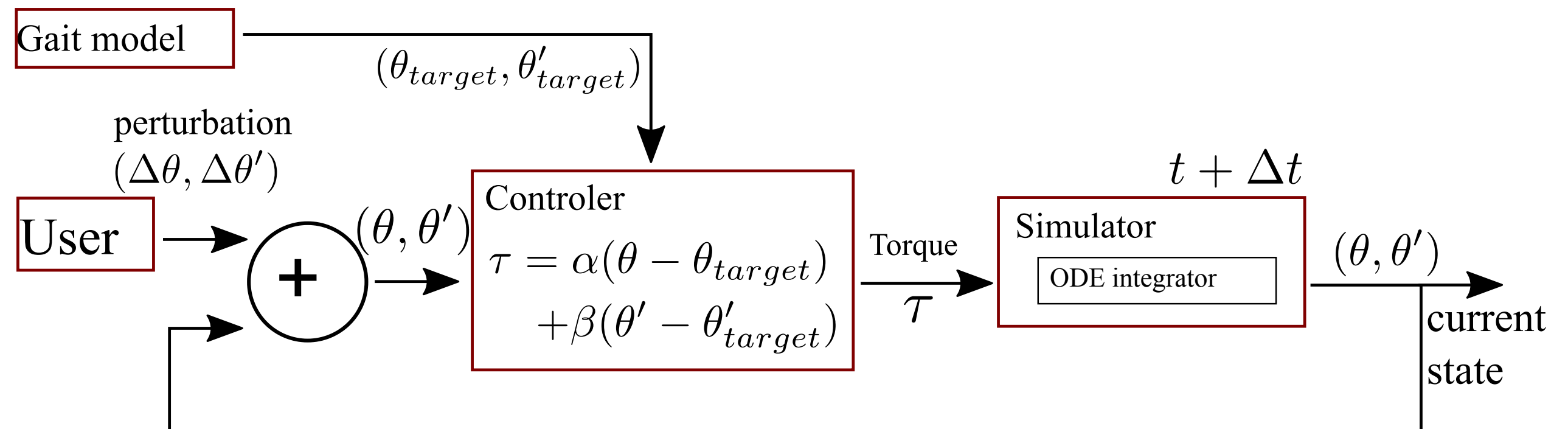
Controllers

Mix between predefined motions and physics → allow user perturbations

- 1 - Define target $(\theta_{target}(t), \theta'_{target}(t))$
pre-defined finite state machine (Gait model)
- 2 - Add user perturbation to the current state
- 3 - Use proportional derivative controllers to compute joint torque τ
- 4 - Integrate torque using rigid body simulator
- 5 - Iterate



Gait model

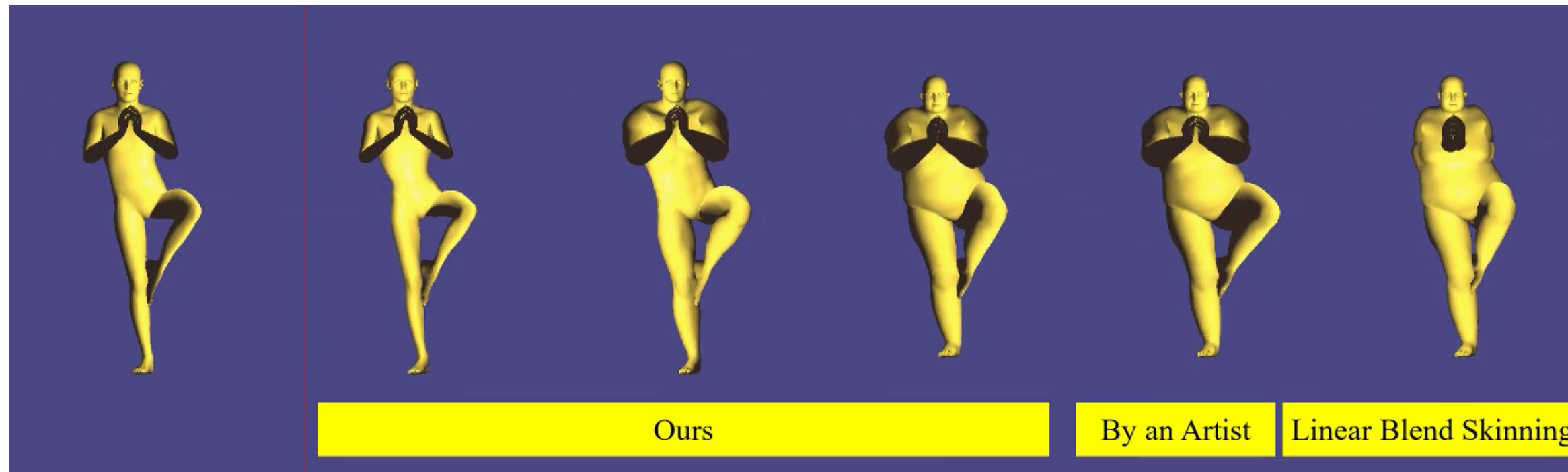
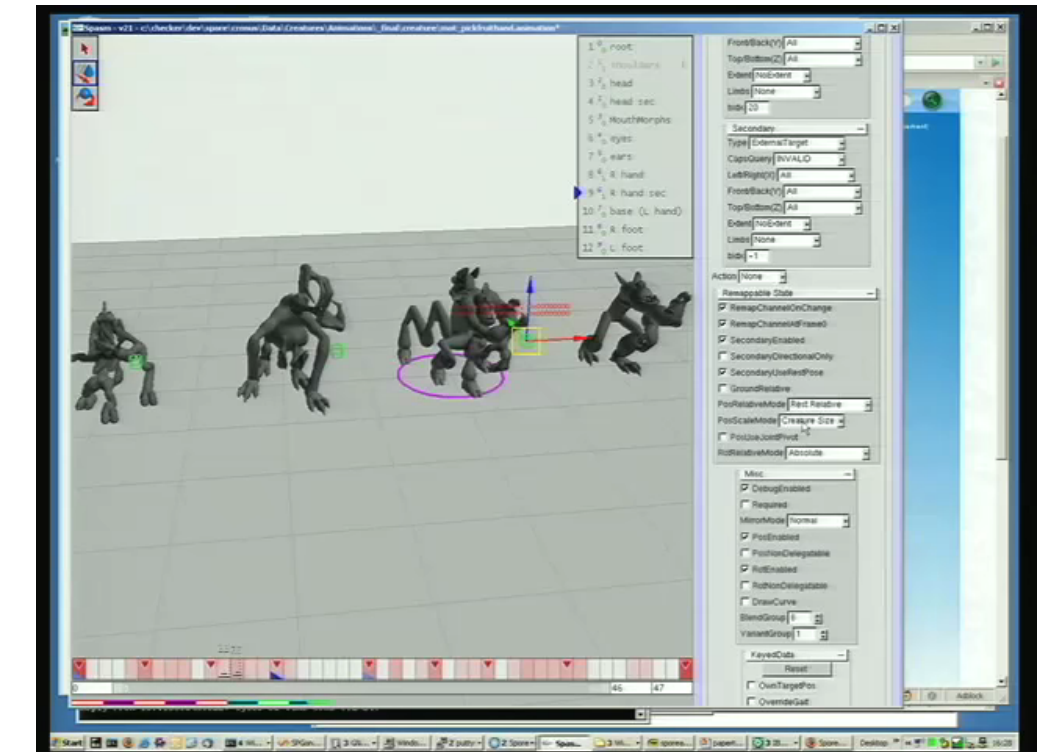


[M. Raibert and J. Hodgins. *Animation of Dynamic Legged Locomotion*, ACM SIGGRAPH 2001]

[K. Yin et al., *SIMBICON: Simple Biped Locomotion Control*, ACM SIGGRAPH 2007]

Motion transfert

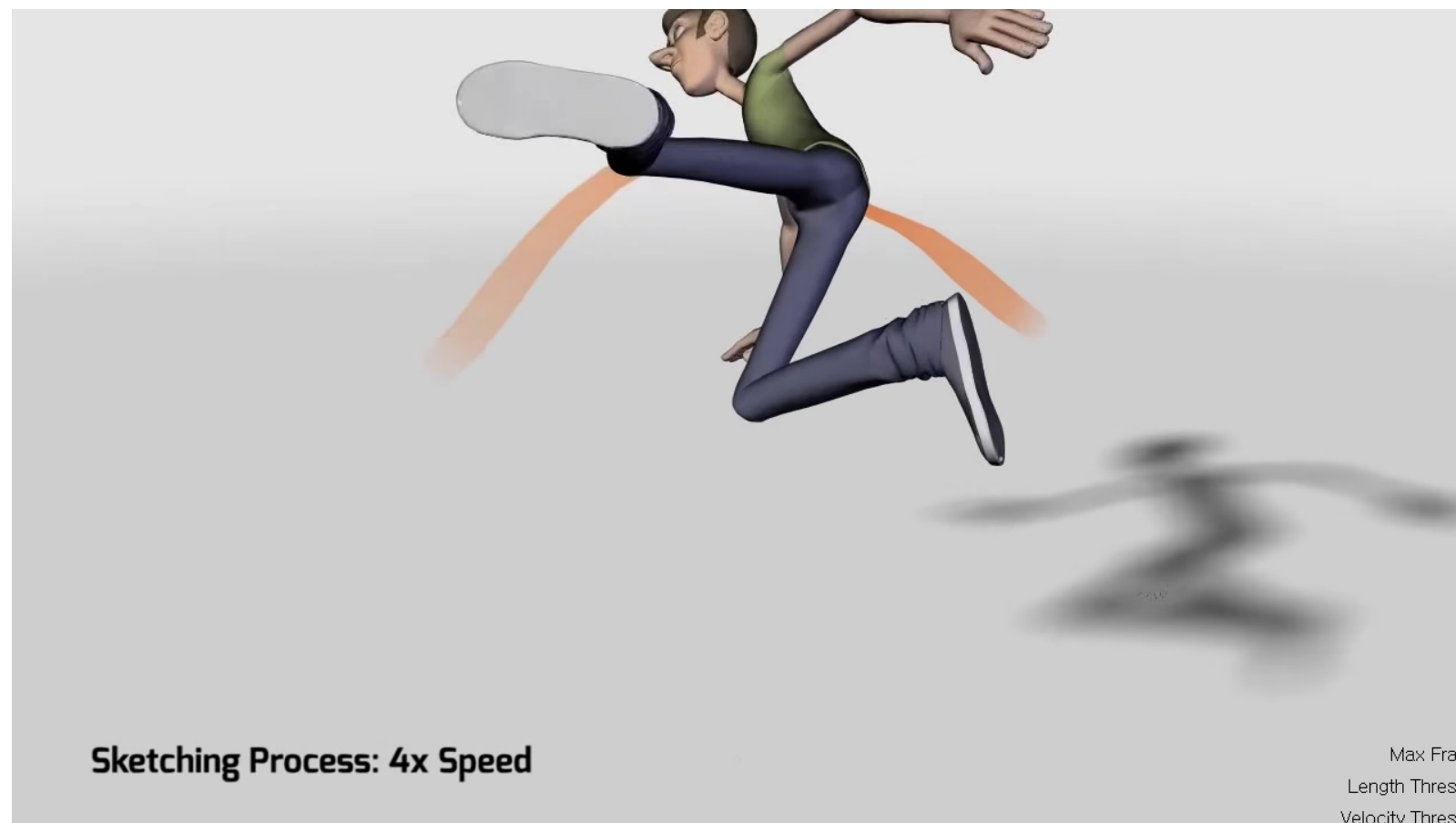
- Local coordinates mapping from skeletal motions
 - [C. Hecker et al., Real-time Motion Retargeting to Highly Varied User-Created Morphologies, SIGGRAPH 2008] (Spore)
- Including shape morphology
 - [Z. Liu et al., Surface based Motion Retargeting by Preserving Spatial Relationship, MIG 2018]



Animation design

Based on the *Line of Action*

- [Guay et al., The Line of Action: an Intuitive Interface for Expressive Character Posing, ACM SIGGRAPH Asia 2013]
- [Guay et al., Space-time sketching of character animation, ACM SIGGRAPH 2015]
- [Choi et al., SketchiMo: Sketch-Based Motion Editing for Articulated Characters, ACM SIGGRAPH 2016]



0:00 / 0:27

0:00 / 0:35

Automatic synthesis of skeletal animation

Seminal works

- [Evolving Virtual Creatures. Karl Sims. SIGGRAPH 1994]
- [Automated Learning of Muscle-Actuated Locomotion Through Control Abstraction. Radek Grzeszczuk and Demetri Terzopoulos. SIGGRAPH 1995]
- *Optimization toward objective function coupled with rigid bodies simulations*
- *Morphological variation from genetic algorithm*

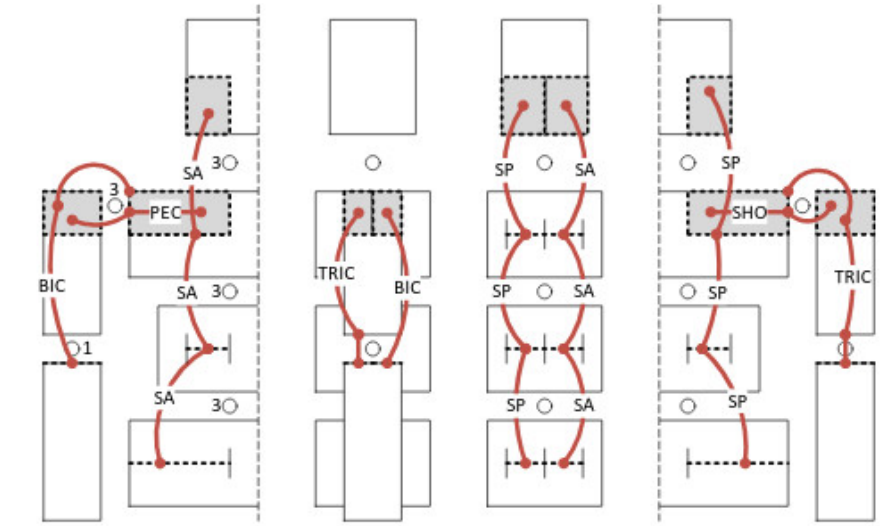
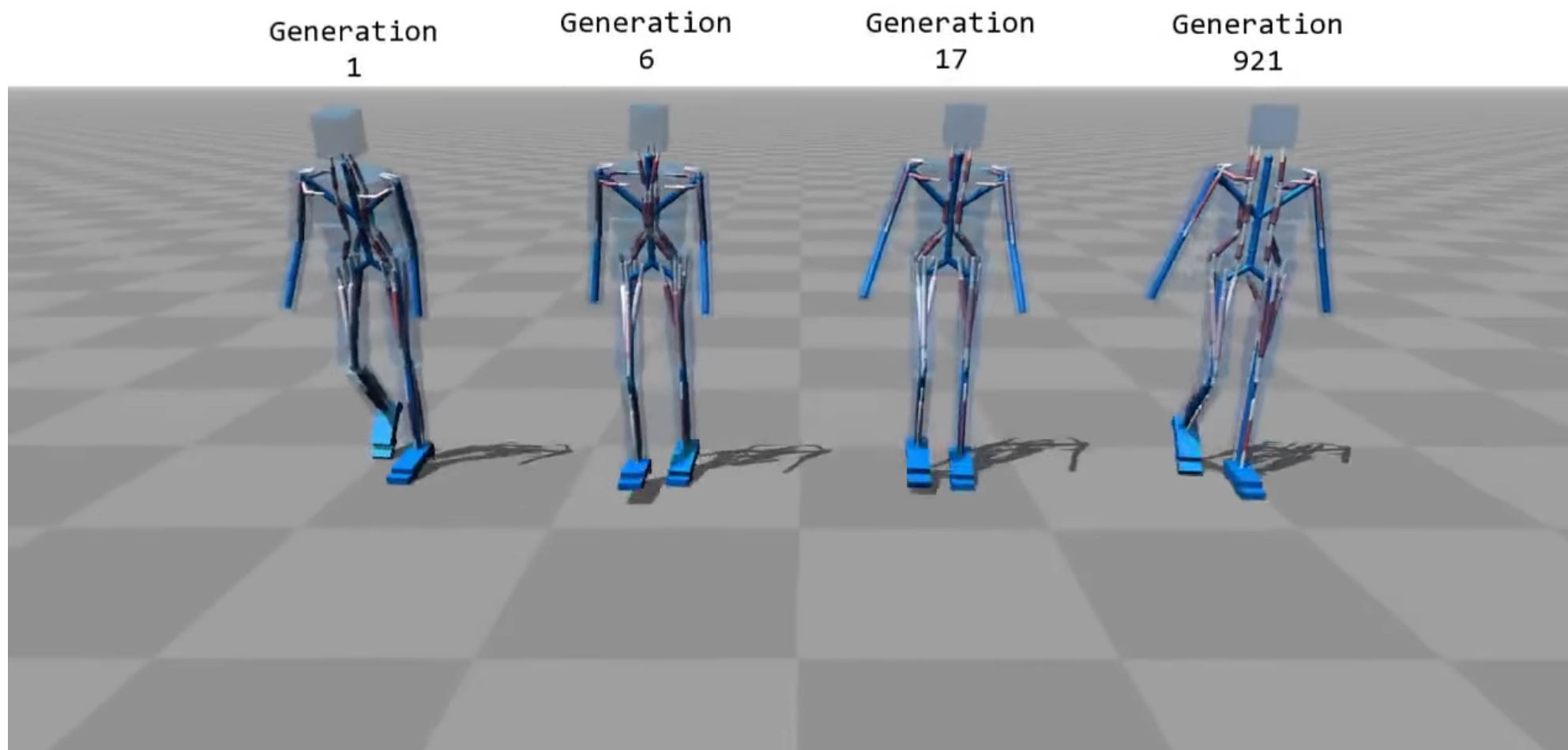
0:00



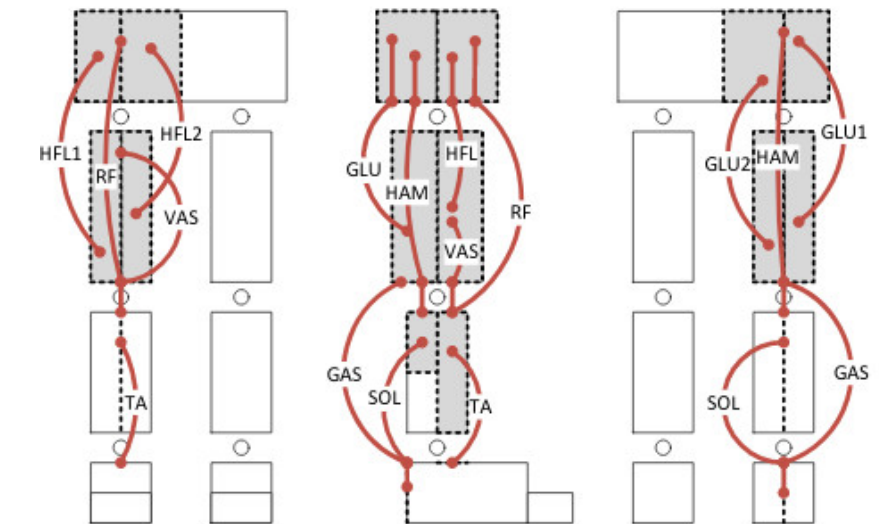
Optimizing muscle activation

- Take into account simple biomechanical model
- Optimize sequence of activation via reinforcement learning

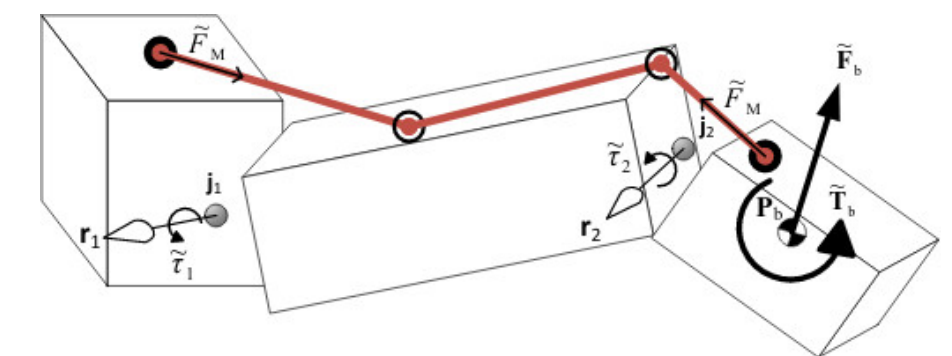
[*Flexible Muscle-Based Locomotion for Bipedal Creatures*, T. Geijtenbeek et al. SIGGRAPH Asia 2013.]



(a) Humanoid upper-body model: front, side arm, side body, and back.



(c) Humanoid lower-body model: front, side and back.

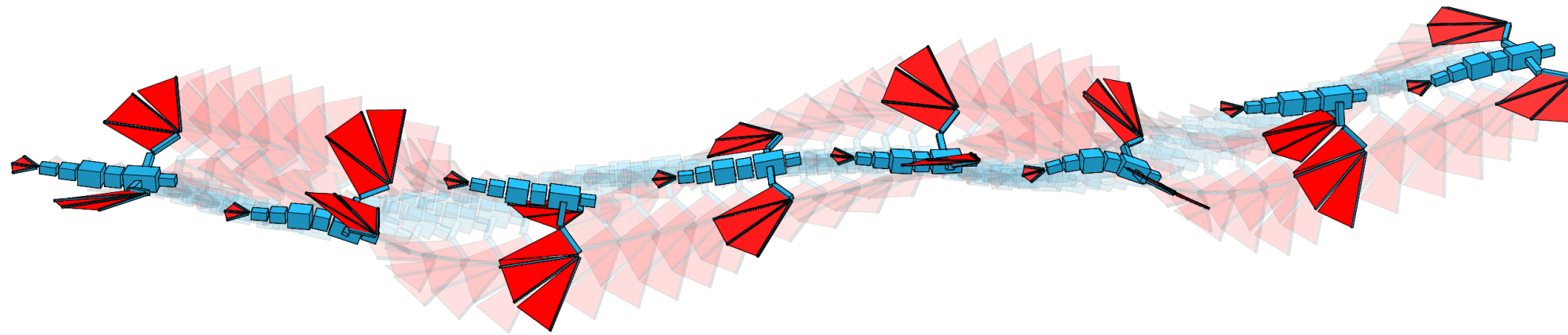


Use of deep learning

Deep reinforcement learning for complex optimization

Learn muscle activation

[Won et al. How to Train Your Dragon: Example-Guided Control of Flapping Flight, ACM SIGGRAPH Asia 2017]



Deep learning for real-time motion control

Learn phase of the motion cycle.

Use large data base of motion capture data

[Holden et al., Phase-Functioned Neural Networks for Character Control, ACM TOG 2017]

