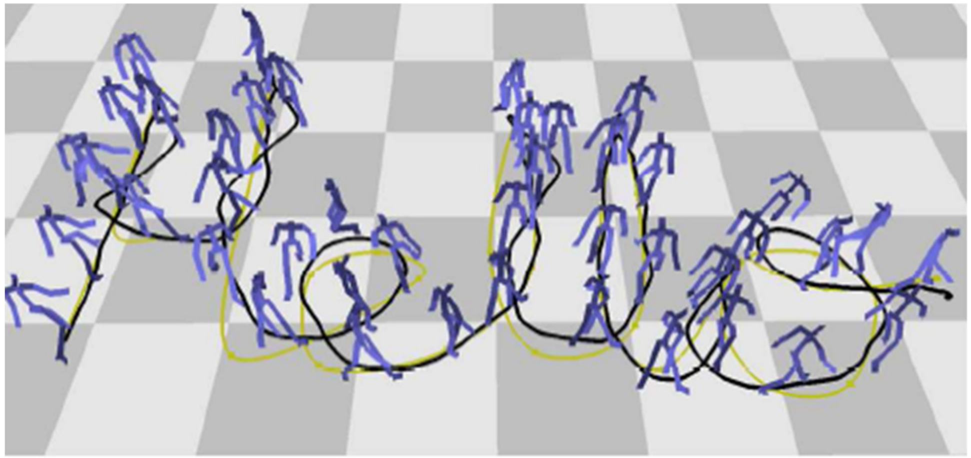
MOTION CAPTURE DATA PROCESSING

- MOTION EDITING / RETARGETING
- MOTION CONTROL / GRAPH
- INVERSE KINEMATIC

Alexandre Meyer

Master Informatique



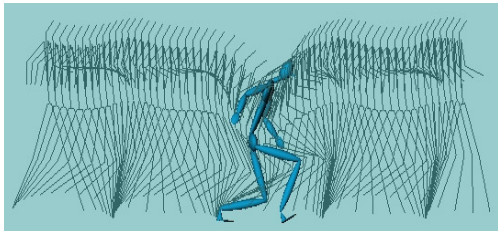
Overview: Motion data processing

In this course

- Motion editing
- Motion blending 2 animations
- Motion FSM/graph
- General Motion blending

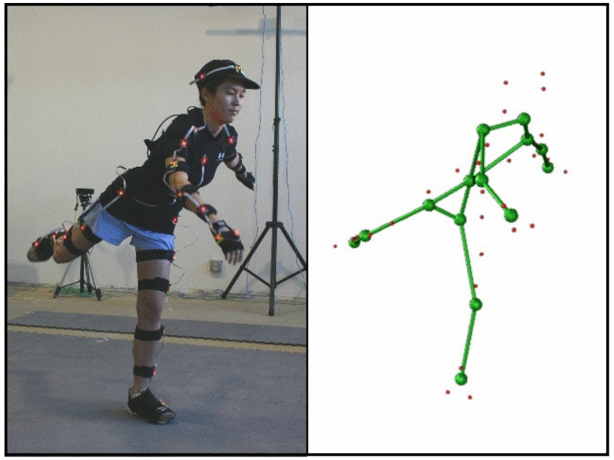
Not in this course

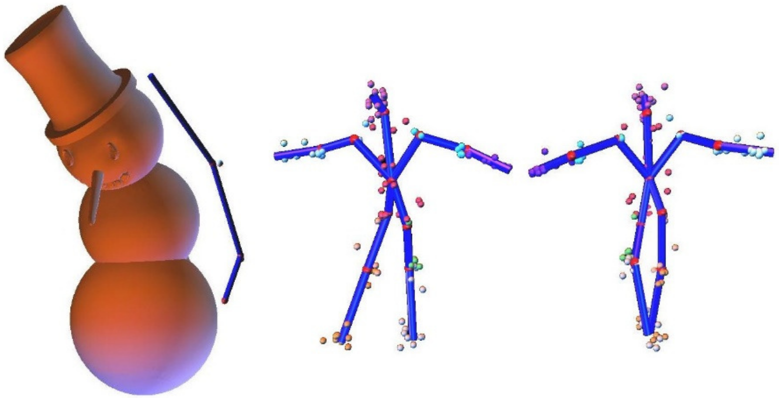
- Motion segmentation
- Motion compression
- Etc.



How do skeletons differ?

- Topology
  - number of bones
  - Connectivity of bones
- Joint Types
  - Bone lengths
  - Anatomical / skin relations
- Is spine in middle of body, or up the back?

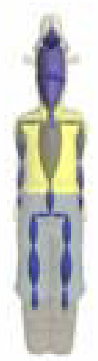


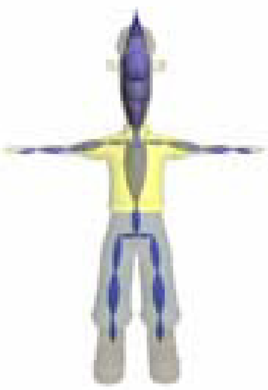


Subtle Skeletal Differences

- Rest Poses (design of a skeleton)
  - Zero Pose / Base Pose
  - Dress or Binding pose
  - Frankenstein Pose
  - Da Vinci Pose
  - Rest Pose (real pose of actor)
- Need to figure out how to get between these

Rest pose



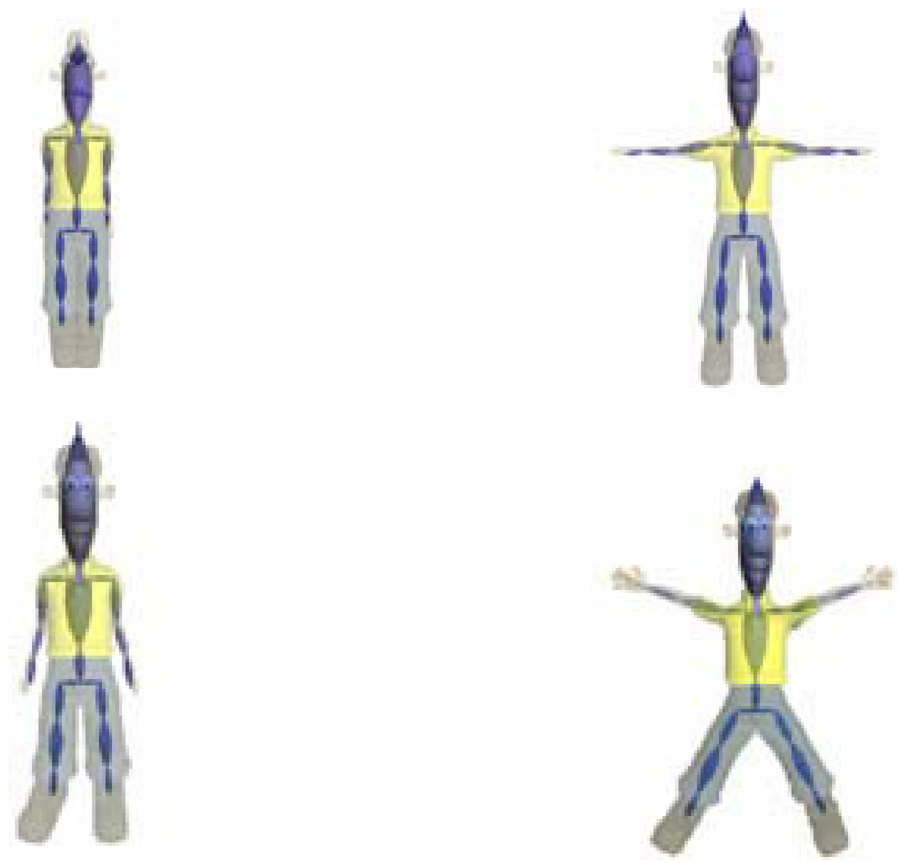


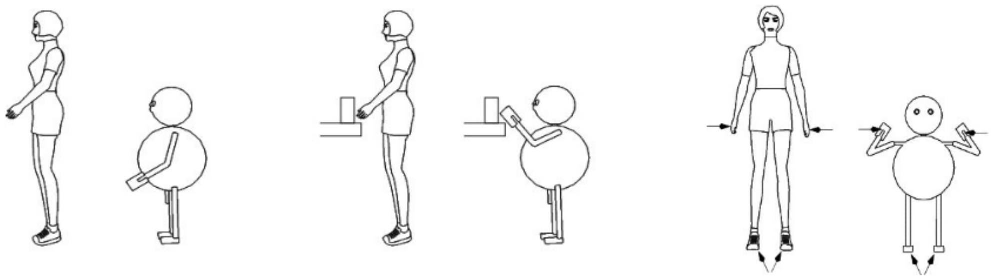
Subtle Skeletal Differences

- Same angles lead to different animation is rest pose is different

Rest pose

Animation with similar angles

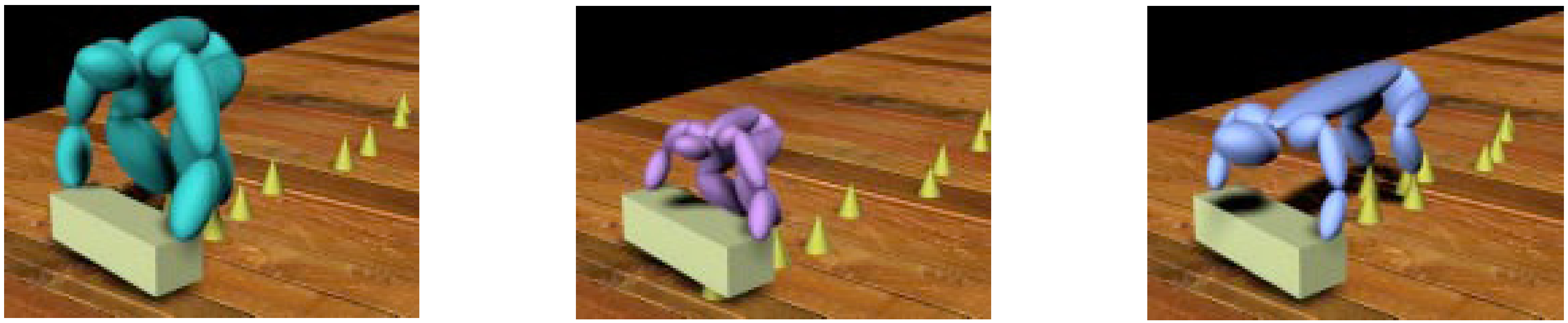




MOTION EDITING

Input: 1 pose of an animation

Edit One Pose: IK, retargeting



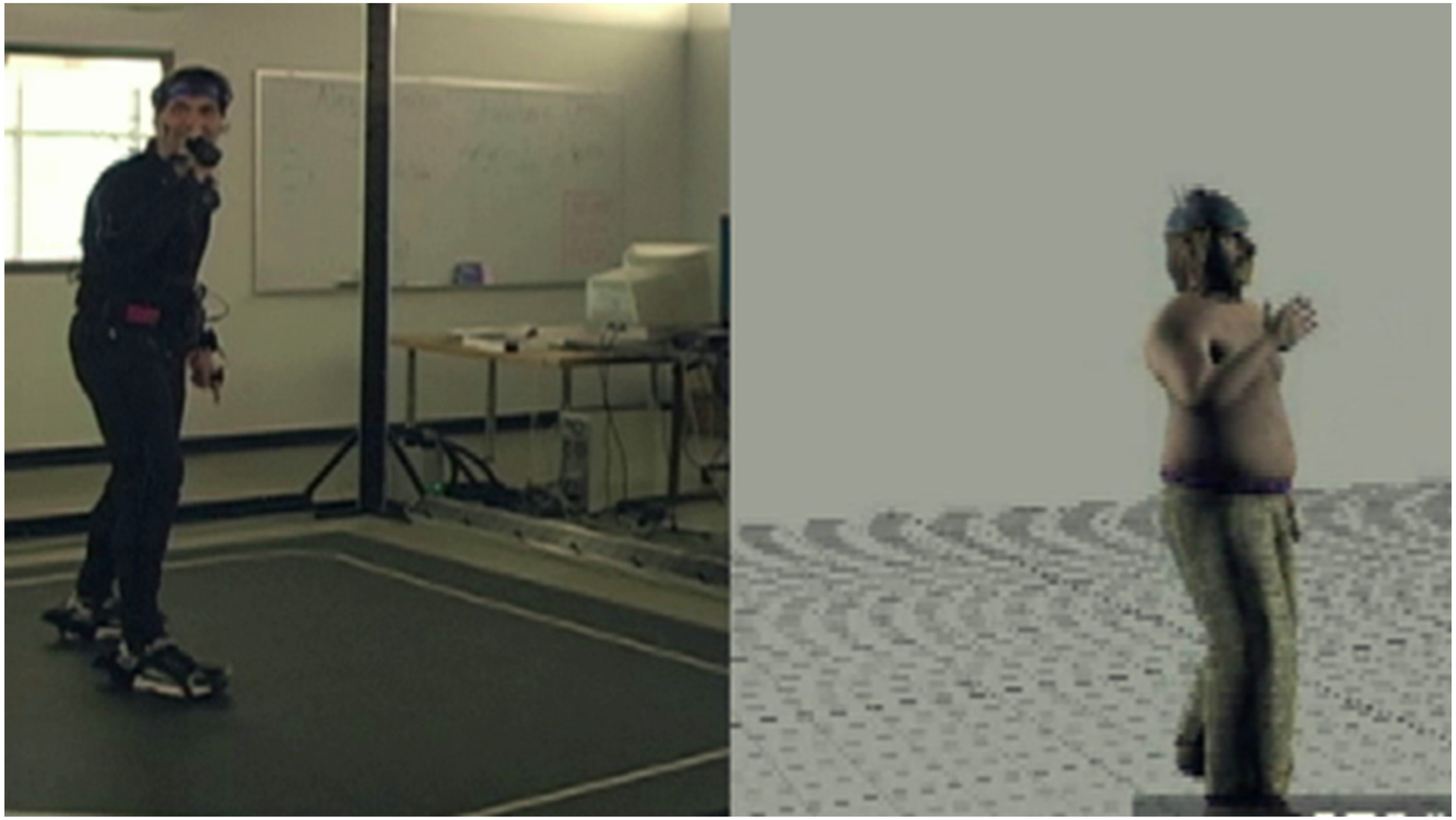
Retargeting

- capture motion on performer
  - **positions** of markers are recorded

- retarget motion on a virtual character
  - motion is usually applied to a skeleton
  - a skeleton is hierarchical
    - linked joints
  - need **rotation** data!

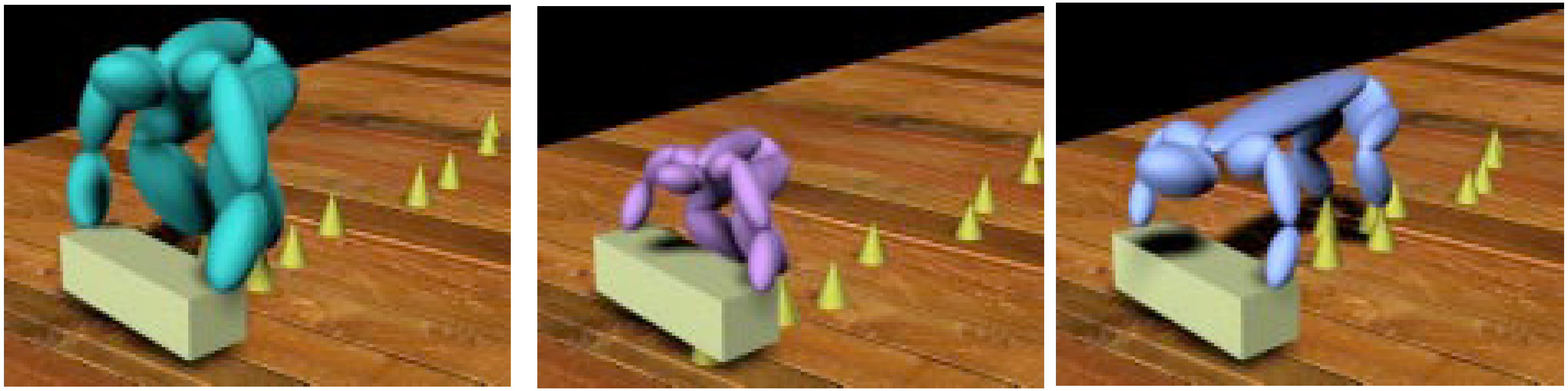
- need to convert positions to rotations

Retargeting problems: hand problem



Problem of Hand or foot position!

- Often hand or foot positions do not match



[Images from Retargetting Motion to New Characters, Gleicher, Siggraph98]

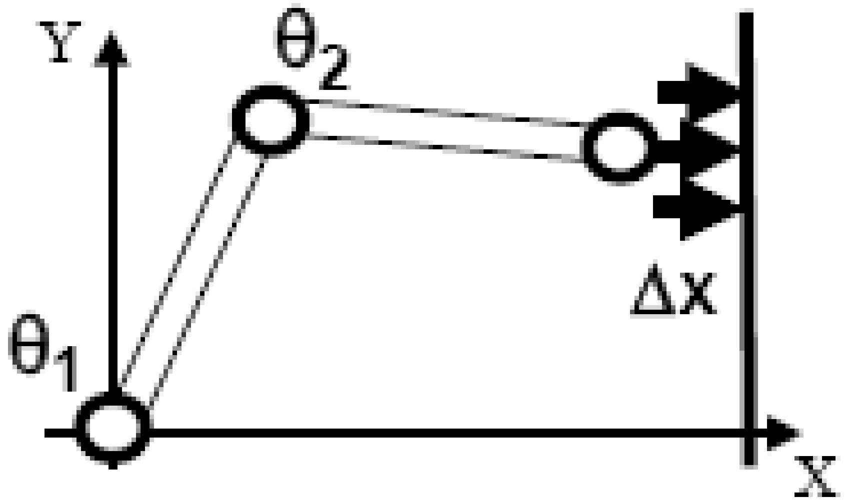
- Need to find a position with hands on the box and feet in concordance with skeleton morphology
- Feet crossing the floor
- **Foot sliding**

➔Quick overview of inverse kinematic

Inverse Kinematics

- Inverse Kinematics
  - Given effectors positions, find a posture(=angles)

- Non-linear problem (position vs. angles)
  - Possibility of no or multiple solutions



Forward Kinematics

- The forward kinematic function f() computes the world space end effector DOFs from the joint DOFs:
  - Forward kinematic is often easy to compute

$$\mathbf{e} = f(\mathbf{\Phi})$$

Inverse Kinematics

- The goal of inverse kinematics is to compute the vector of joint DOFs that will cause the end effector to reach some desired goal state
- In other words, it is the inverse of the forward kinematics problem
  - $f^{-1}()$ usually isn't easy to compute

$$\mathbf{\Phi} = f^{-1}(\mathbf{e})$$

Inverse Kinematics

Inverse Kinematics: many approaches

- Analytic method [IKAN, Badler]
  - Geometric based, fast
  - Ok only for few joints
- Numeric solution
  - Iterative process
  - Expensive
  - Flexible (constraints)
  - Minimization problem

Editing One Pose

See the course on IK

MOTION EDITING

Input: 1 animation

The General Challenge

What you get is not what you want!

- You get observations of the performance
  - A specific performer
  - A real human
  - Doing whatever they did
  - With the noise and "realism" of real sensors
- Want something else
  - But need to preserve original
  - But we don't know what to preserve
  - Can't characterize motion well enough

Three Problems

- Where does X live in the data?
  - Where X ∈ {style, personality, emotion, …}
  - The things to keep or add

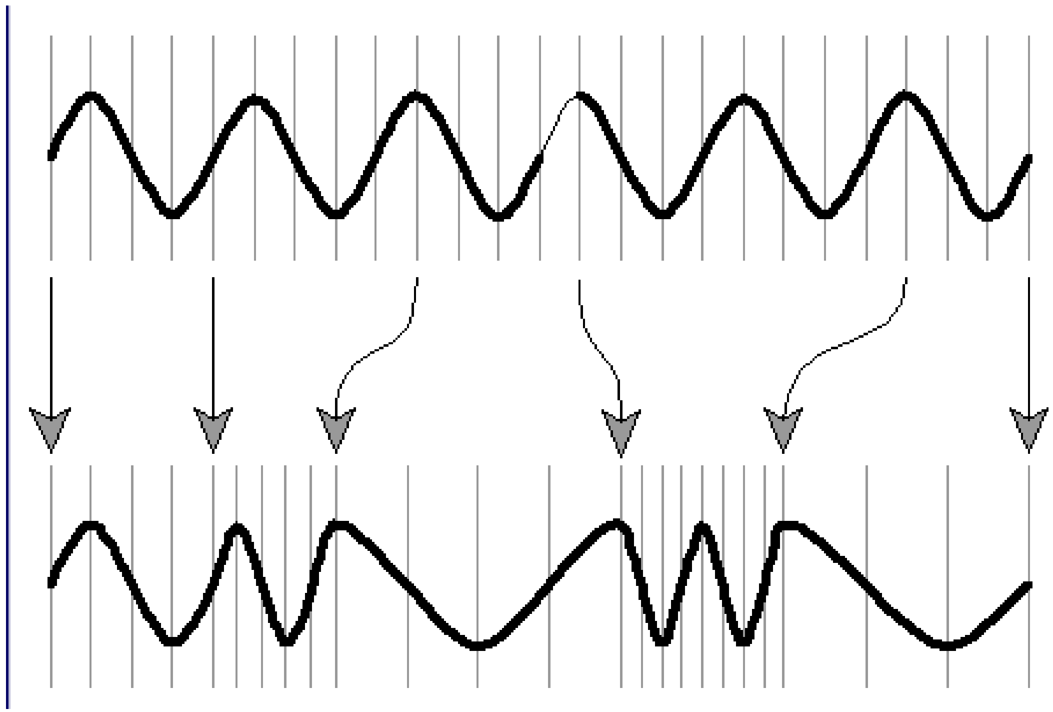
- Small artifacts can destroy realism
  - Eye is sensitive to certain details

- How to *specify what you want ?*

Manipulating motion

- Manipulate time: Motion slower or faster
  - m(t) = m0(f(t))
  - f : R - > R “time warp”
- Time scaling
  - f(t) = k t
- Time shifting
  - f(t) = t + k
- Time warping
  - Interpolate a table
  - Align events

VIDEO

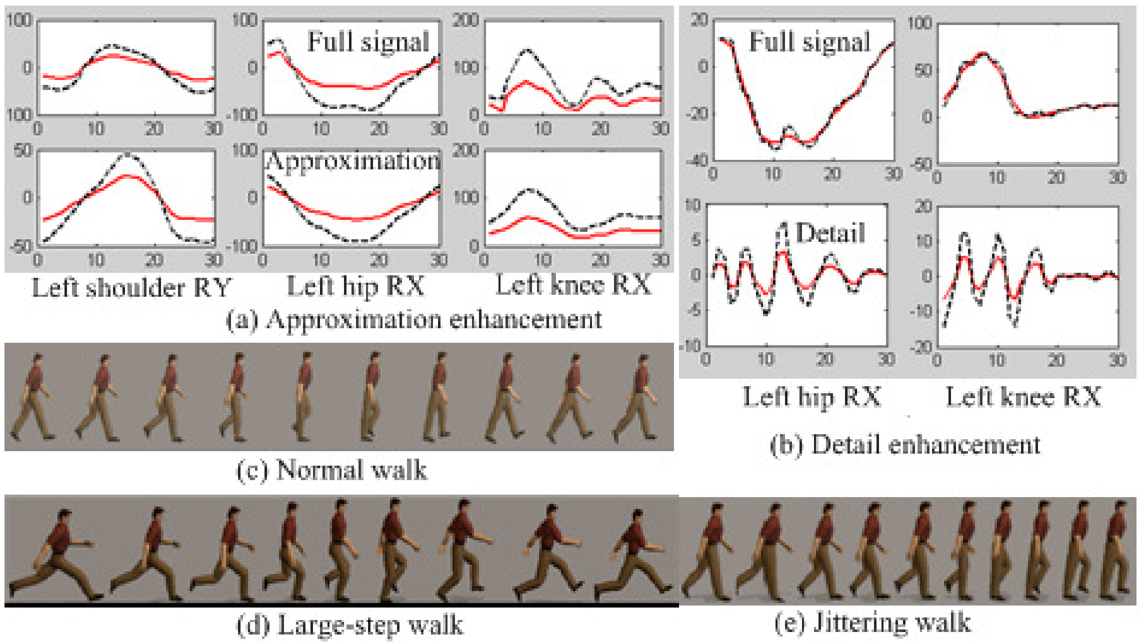


Manipulating motion

- Manipulate value
  - m(t) = f(m0(t))
  - f : Rn- > Rn
- Scale?
  - For instance each angles x 2 → Exagerate motion
- Shift?
- Convole (linear filter)
- “Add” to another motion
  - $m(t) = m_0(t) + a(t)$

Noise Removal: Signal Processing

- Noise comes from errors in process
  - Sensor errors
  - Fitting errors
  - Bad movements
- Nose is “data” that we don’t want



Where's the Noise?

- Sometimes identification is easy
  - Clearly wrong (foot through floor)
  - Marked wrong (missing data - gaps)
- More often, need to guess
  - Might be a subtle twitch…
  - Might be person shaking…
  - Might be sensor errors…

➔simply apply a filter ?

Important Intuition

- High Frequencies are Important!
- Always significant
  - Impact
  - Rapid, sudden movement
  - …

Signal processing [Unuma95]

- Fourier series
  - Coefficient motion parameters (emotion, gait)

Exaggerate jump by scaling low frequency

Motion Signal Processing [Bruderlin95]

- Foreach channel of each joint

Input signal N values									
10	14	16	20	22	24	28	22	…	…

N/2 values				
12 (± 2)	18 (± 2)	23 (± 1)	25 (± 3)	…

~high frequency

N/4 values		
15 (± 3)	24 (± 1)	…

…

1 value
19.5 (± 4.5)

~low frequency

G = white values; L=red values

Motion Signal Processing [Bruderlin95]

- G : (left) white value of previous slide
- L : (right) red value of previous slide

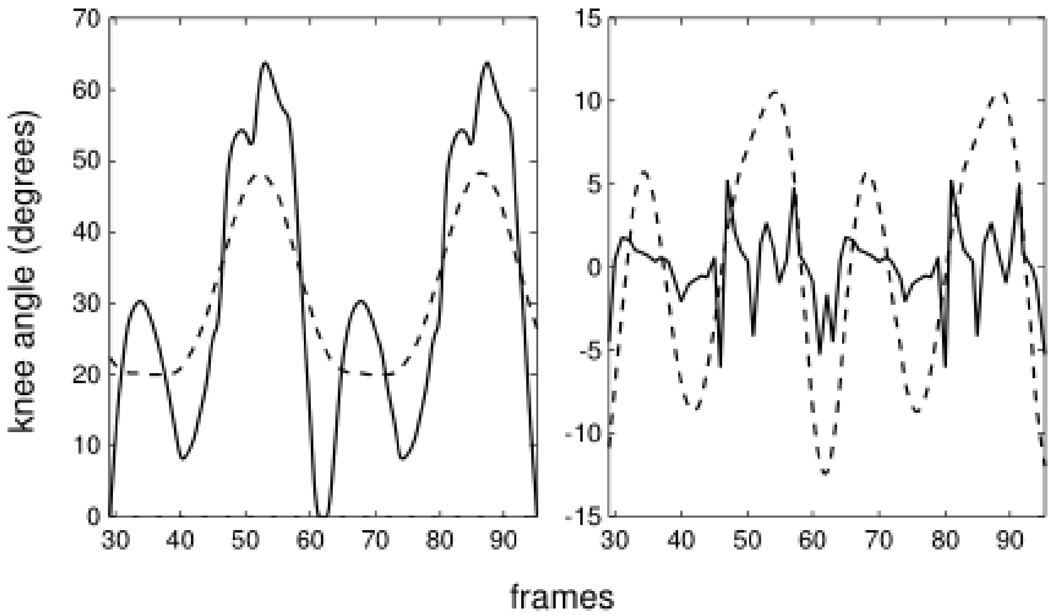
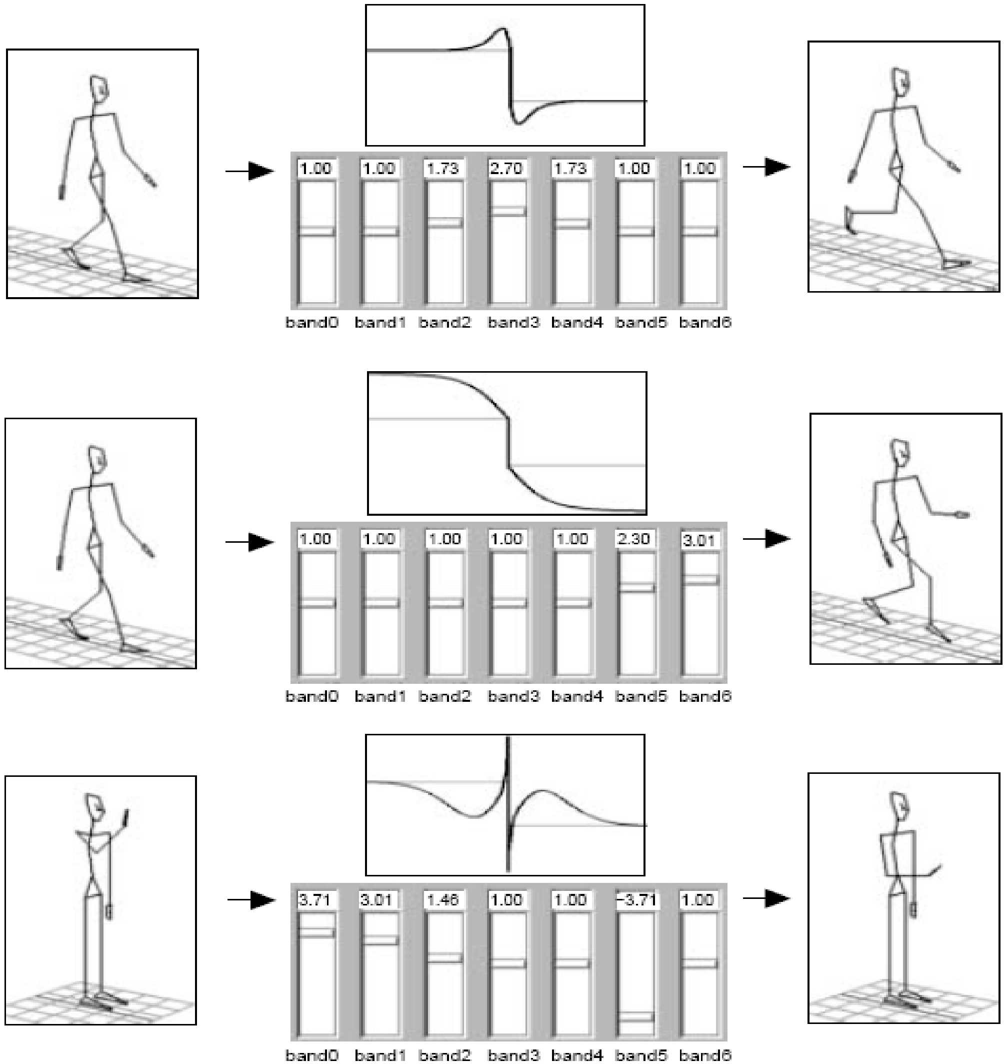


Figure 2: Left: lowpass G_0 (solid) and G_3 (dashed; B-spline kernel of width 5); right: bandpass L_0 (solid) and L_2 (dashed) of the sagittal knee angle for two walking cycles.

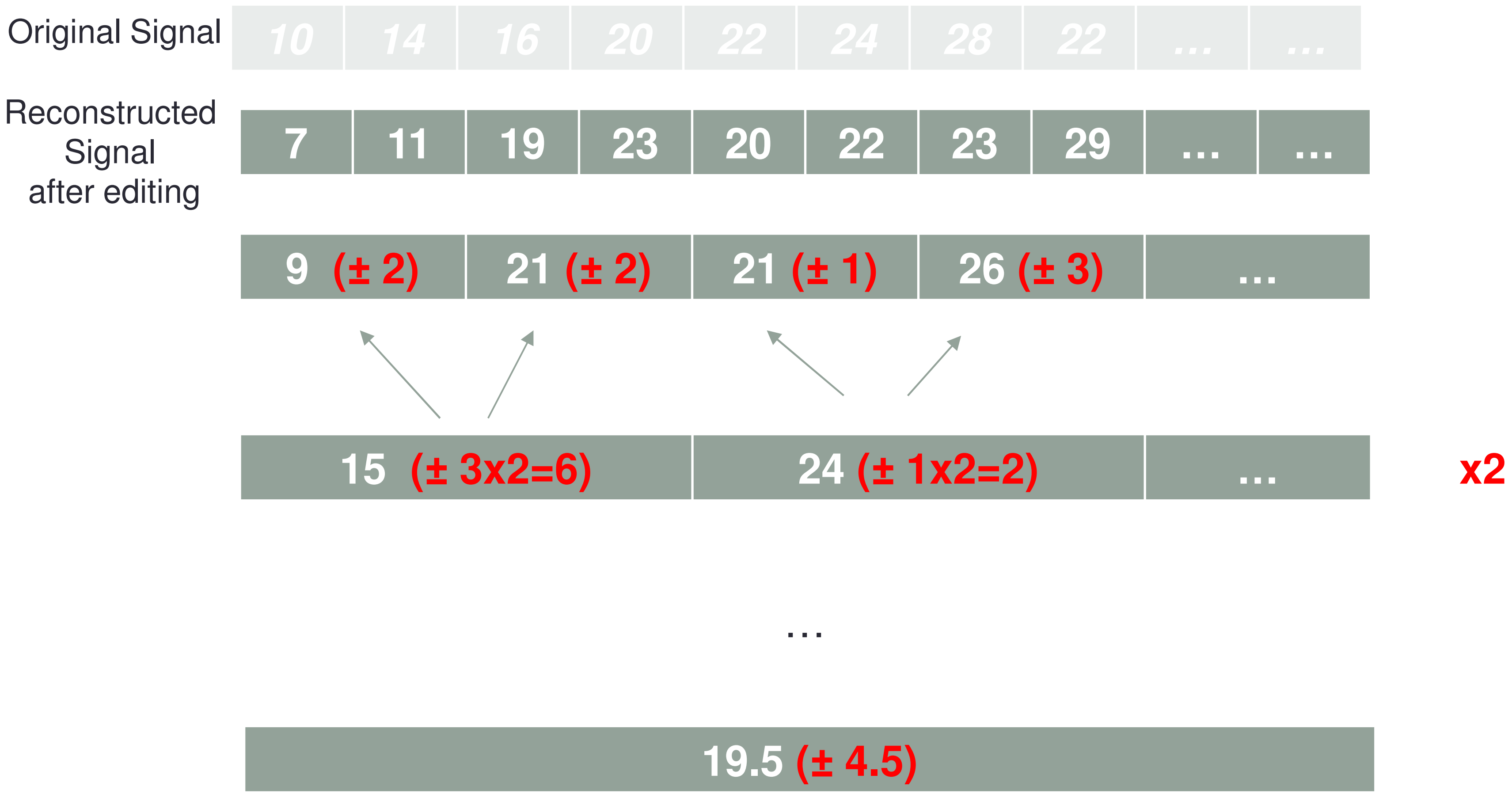
Motion Signal Processing [Bruderlin95]

Siggraph95



Motion Signal Processing [Bruderlin95]

- Edit white values and red values with slider (multiplication)



G = white values; L=red values

Signal processing and style [YM2016]

- Yumer M.E. and Mitra N.J., Spectral Style Transfer for Human Motion btw. Independent Actions, SIGGRAPH 2016.

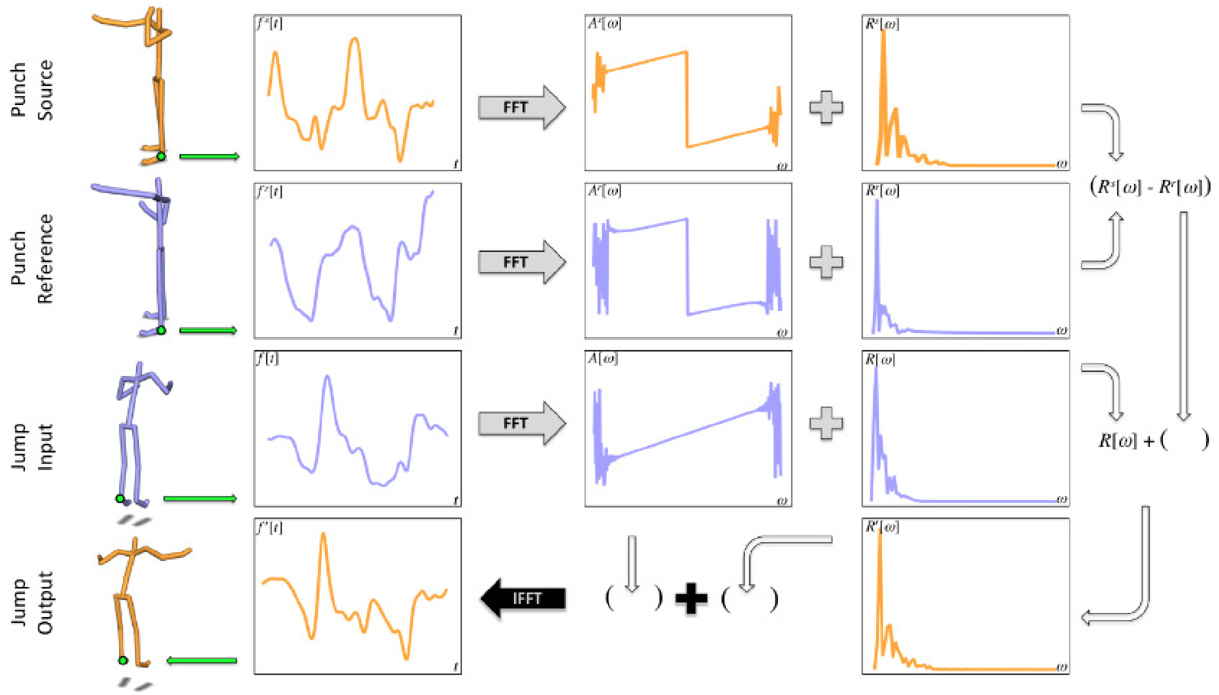
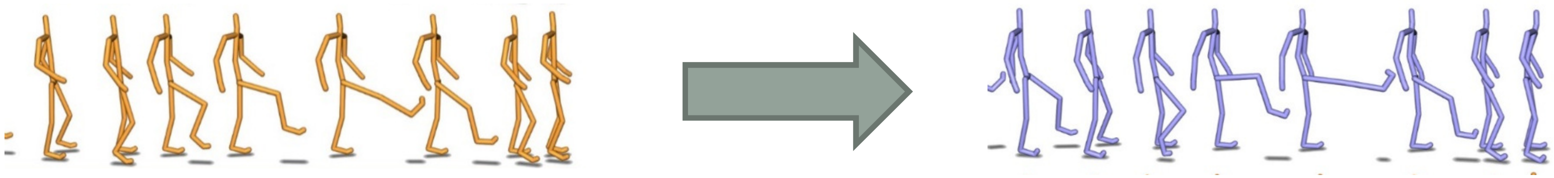


Figure 4: *Time domain signals: target $f[t]$, source $f^s[t]$, and reference $f^r[t]$.Spectral domain processing: we keep $A[\omega]$ constant, and apply the difference of $R^s[\omega]$ and $R^r[\omega]$ to $R[\omega]$ under real-only time-domain signal constraint to compute $R'[\omega]$. Stylized magnitude $R'[\omega]$ and constant $A[\omega]$ result in the stylized time domain data.*

VIDEO

MOTION EDITING

Transition between 2 Animations

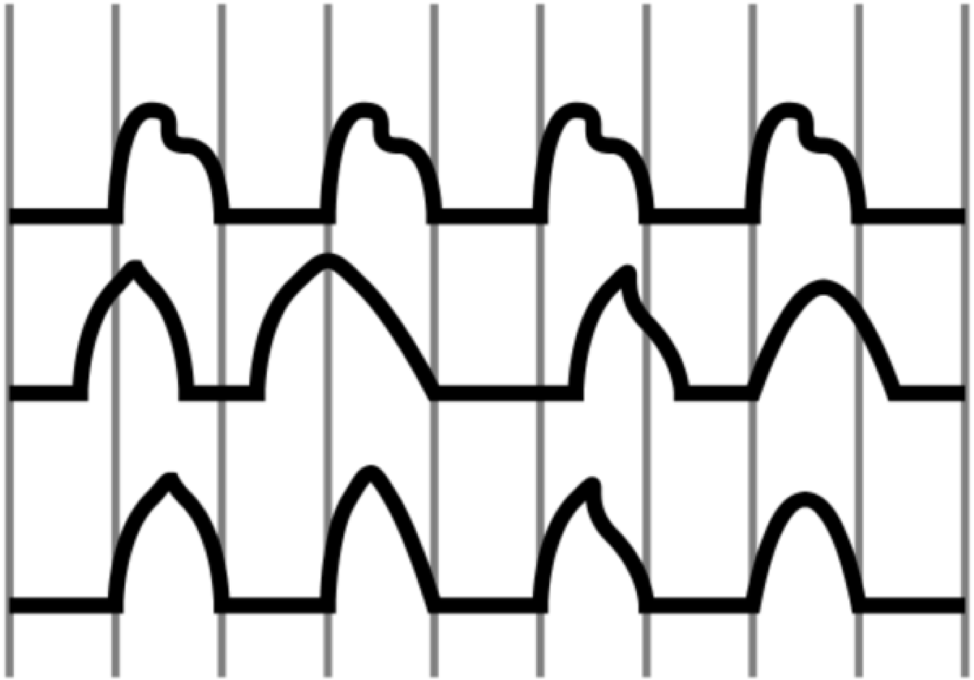


Motion Blending (interpolation)

- “Add” two motions together
  - Really interpolate
- $m(t) = a\ m0(t) + (1-a)\ m1(t)$
  - Note: this is a per-frame operation
- It works only if poses are similar!!!
- Very useful!
  - Often get small pieces of motion
  - Need to connect
  - Easy if motions are similar

Motion Blending

- “Add” or "blend" two motions together
  - Works only if motion are synchronized



Motion Blending

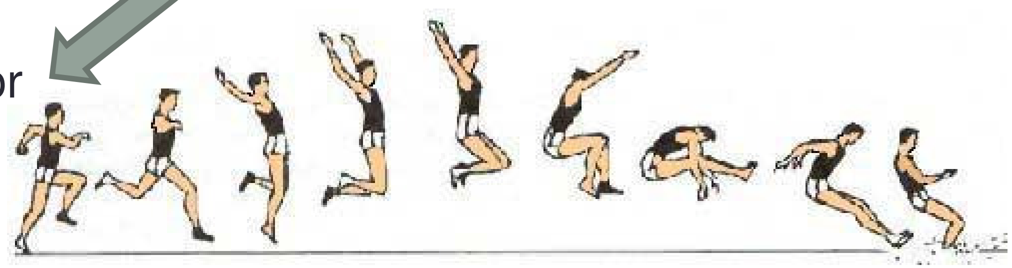
Anim 1

Right leg on the floor

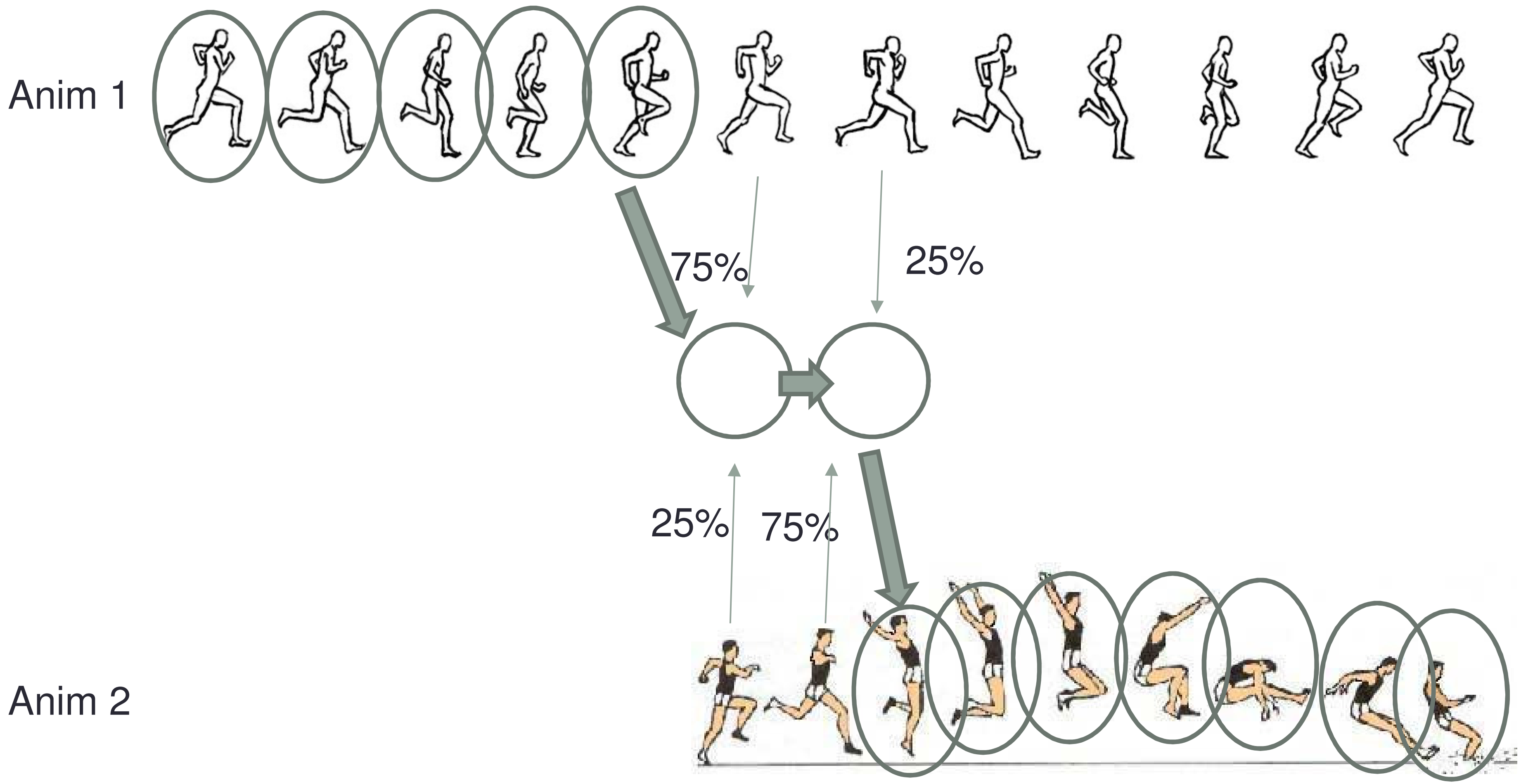
Bad transition

Left leg on the floor

Anim 2



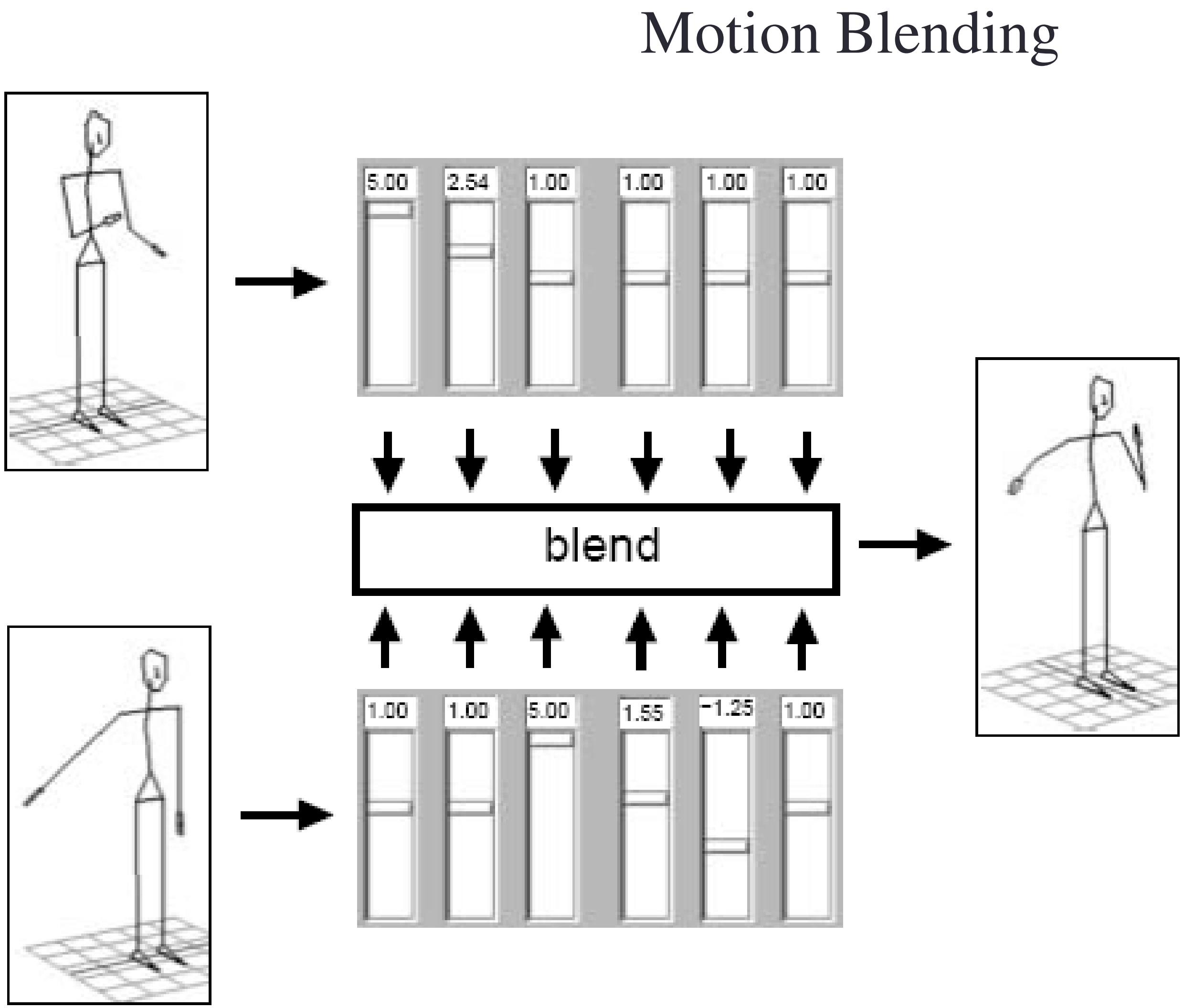
Motion Blending : good transition



Motion Blending using Signal Processing

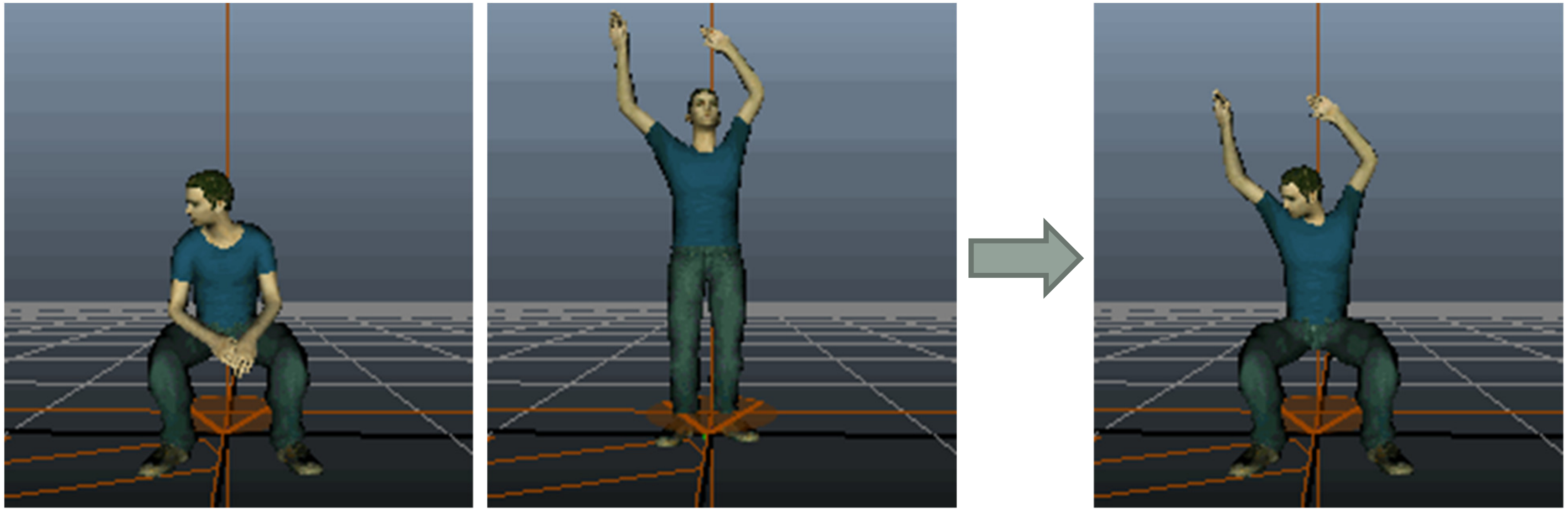
[Bruderlin95]

Siggraph95

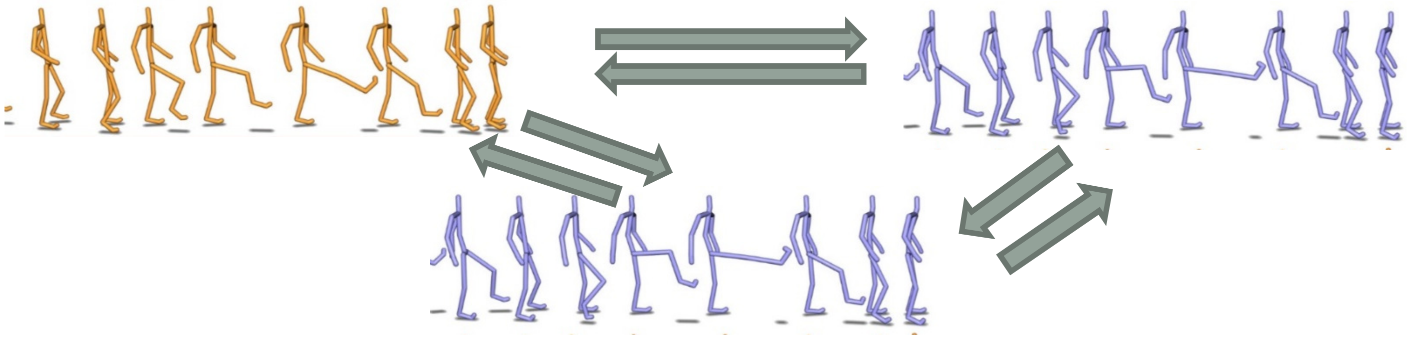


Motion Blend by Body Parts

- Combine different motions for each body parts



http://golaem.com



MOTION EDITING

Input : set of N animations

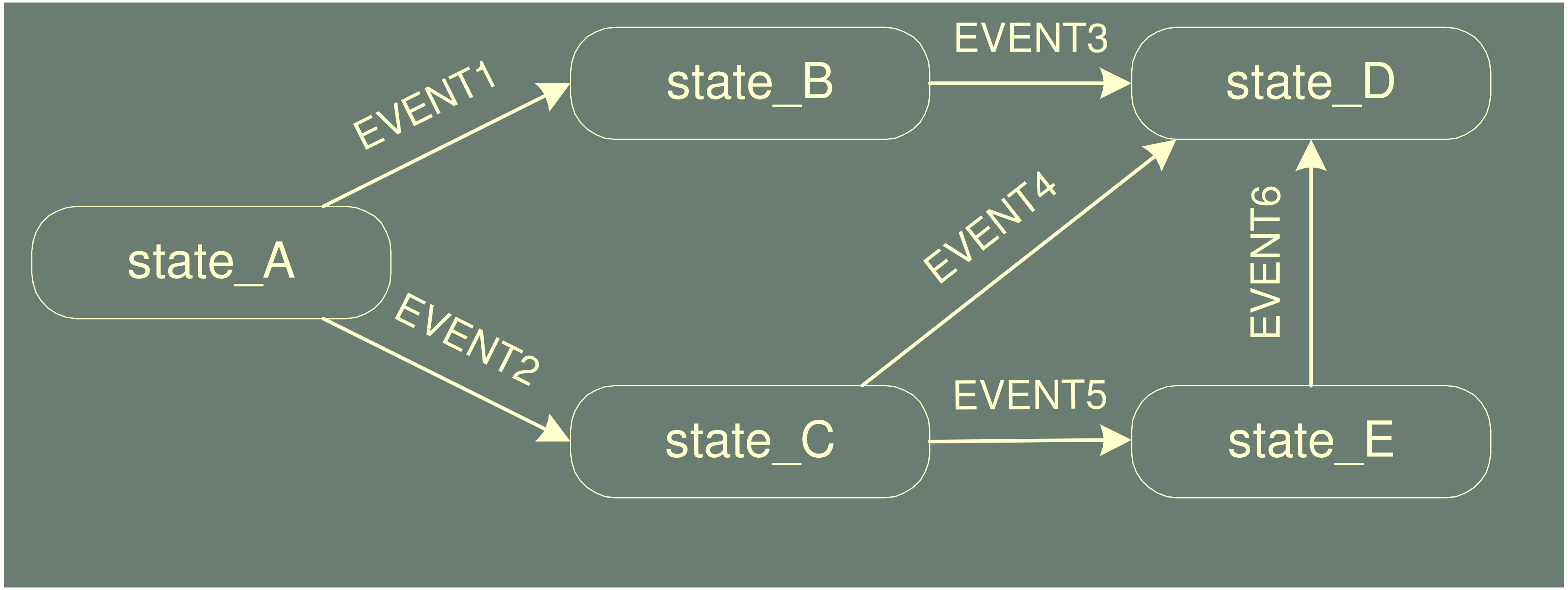
➔Character Control with

- Finite State Machine
- Motion Graph

Finite State Machines

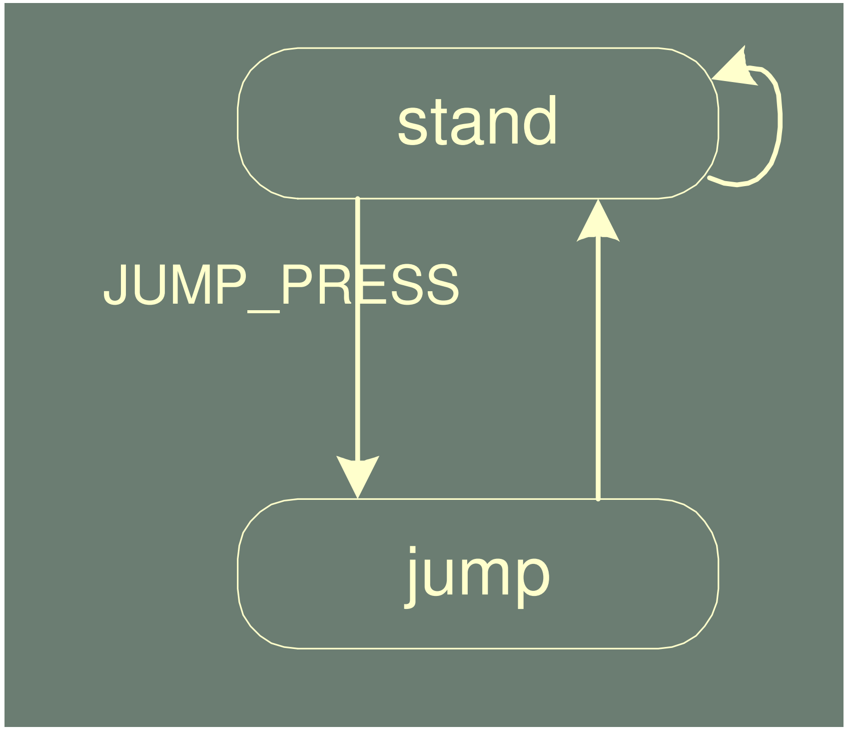
- States represent animations
- Transitions represent instantaneous events
- Transitions can be triggered by
  - End of animation
  - Button press
  - In-game event (collision…)
  - Timers
  - Whatever…
- State machines can be blended. Blenders can be controlled by state machines…

State Machines

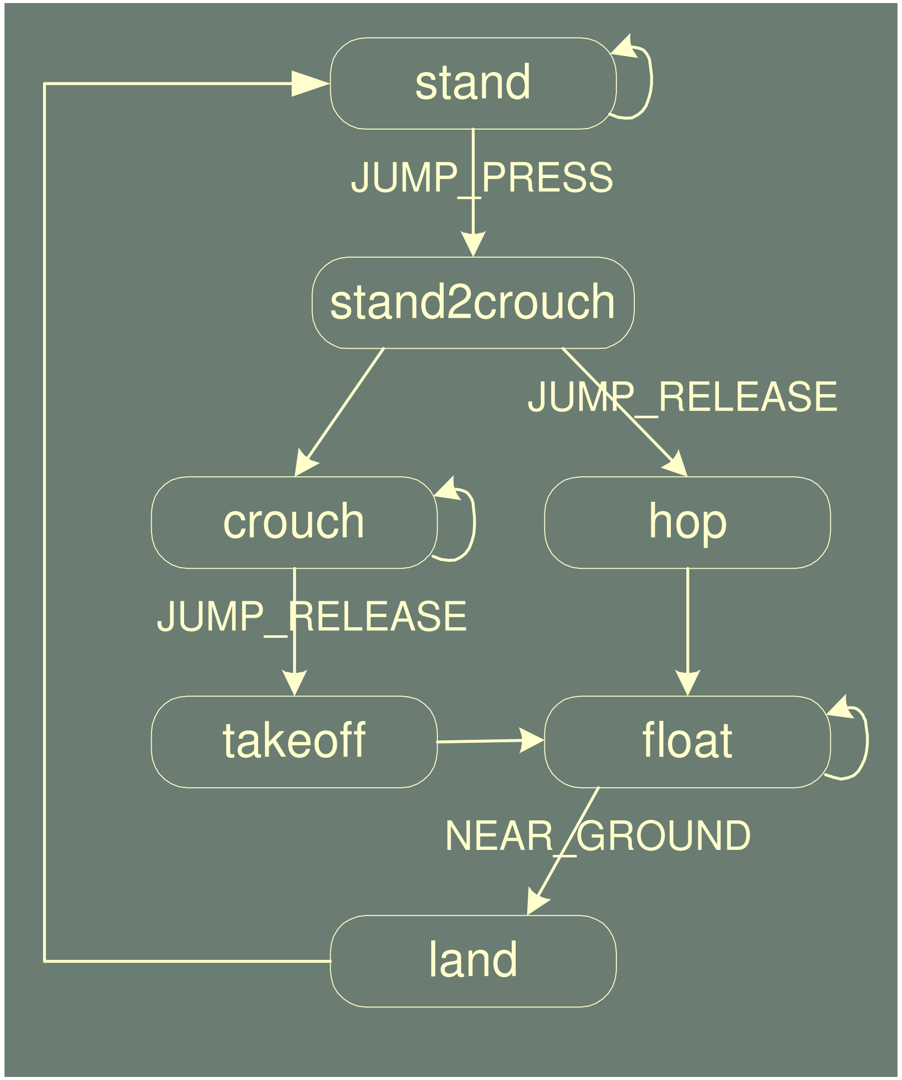


Simple Jump State Machine

- Consider a simple state machine where a character jumps upon receiving a JUMP_PRESS message



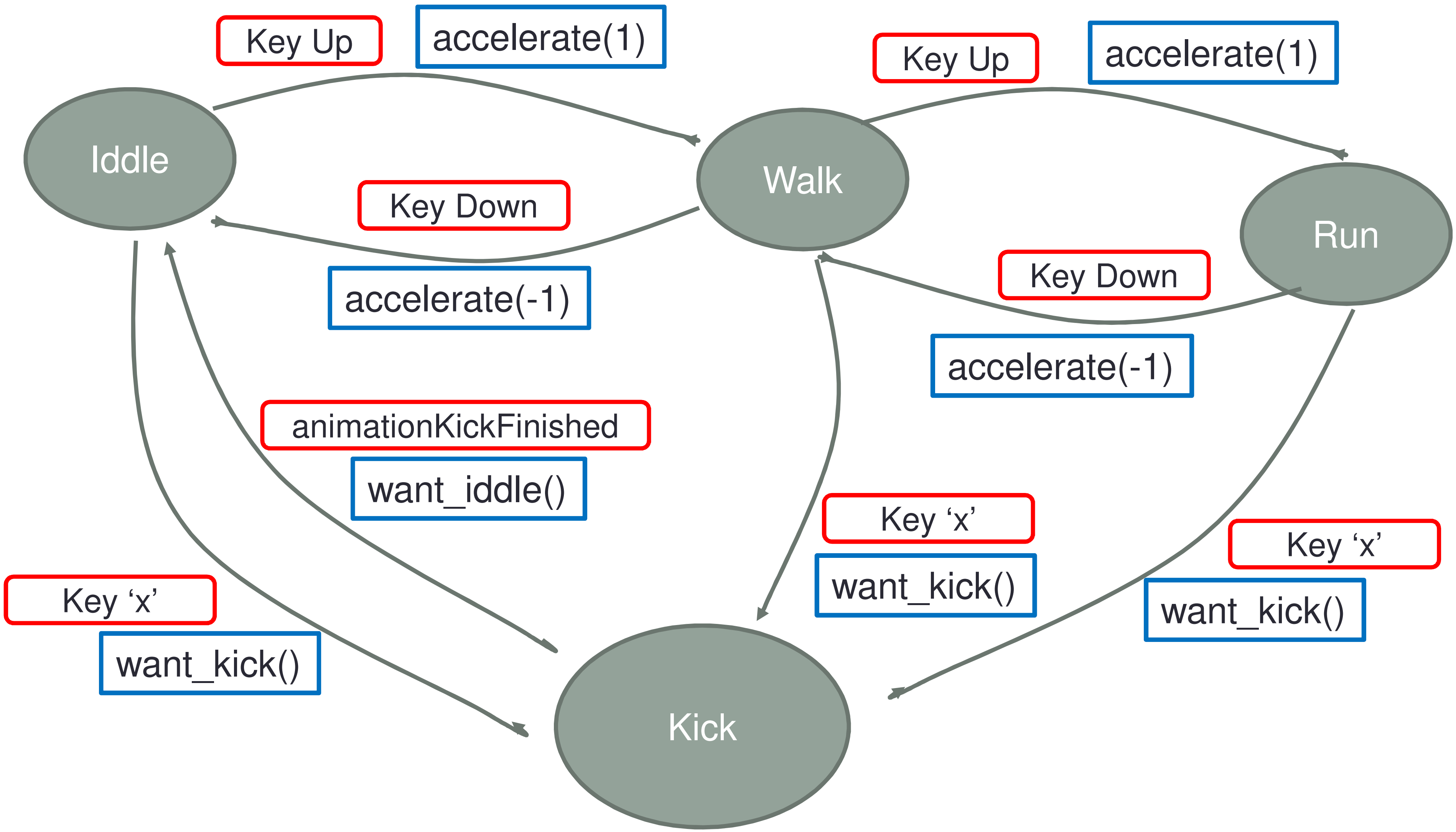
More Complex Jump



State Machine (Text Version)

STATE	EVENT	ACTION
stand	{JUMP_PRESS	stand2crouch }
stand2crouch {		
	JUMP_RELEASE	hop
	END	crouch }
crouch	{JUMP_RELEASE	takeoff }
takeoff	{END	float }
hop	{END	float }
float	{NEAR_GROUND	land }
land	{END	stand }

State Machine (example)

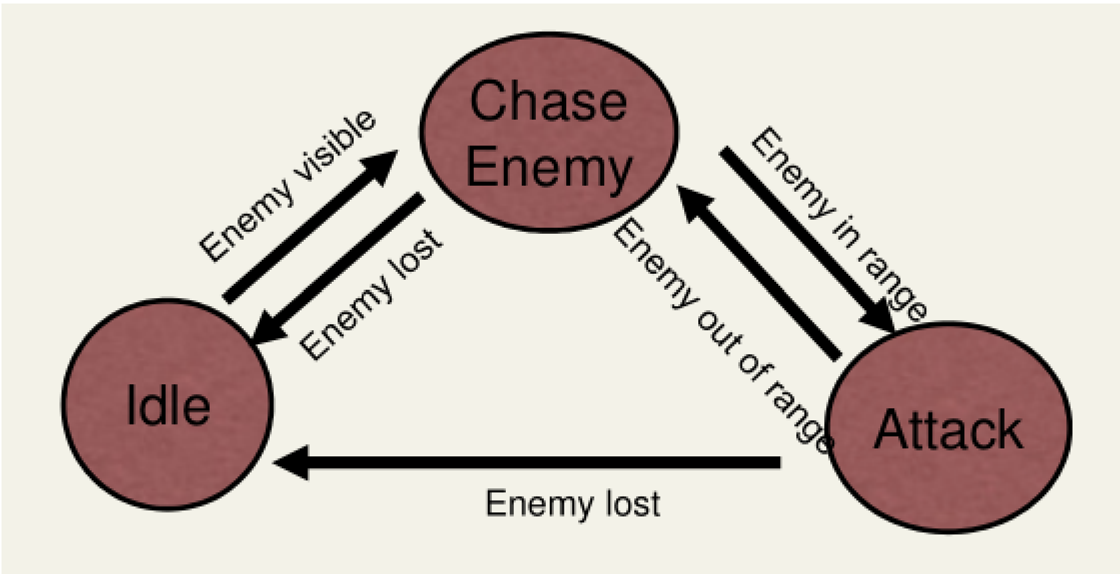


State Machines and IA

- FSM ➔ Behaviour ➔ Play Motion Capture Animation

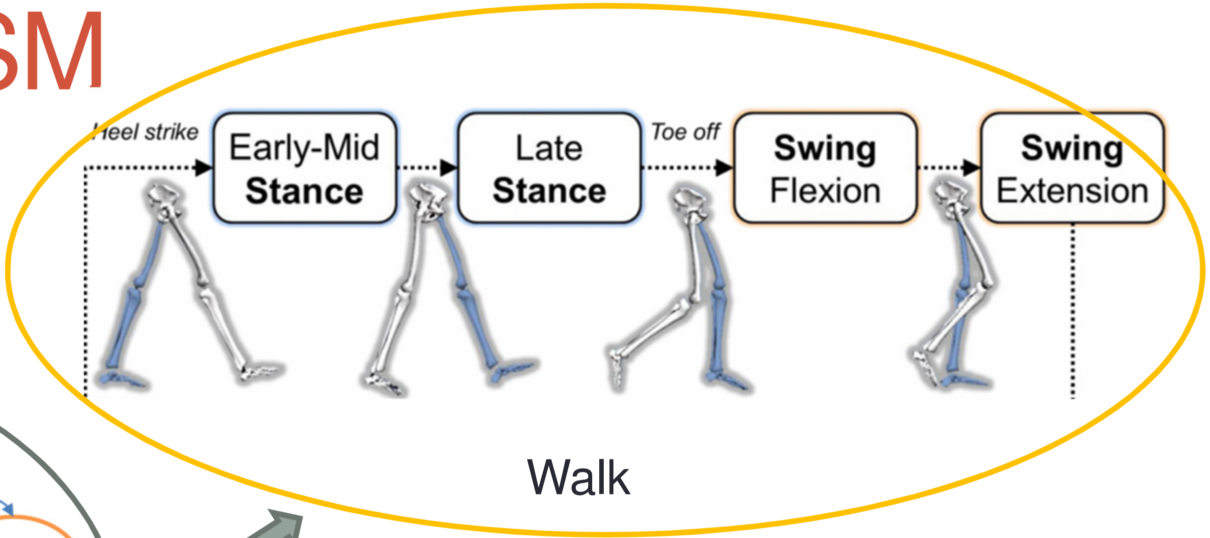
IA — Computer Animation

- First person shooter example

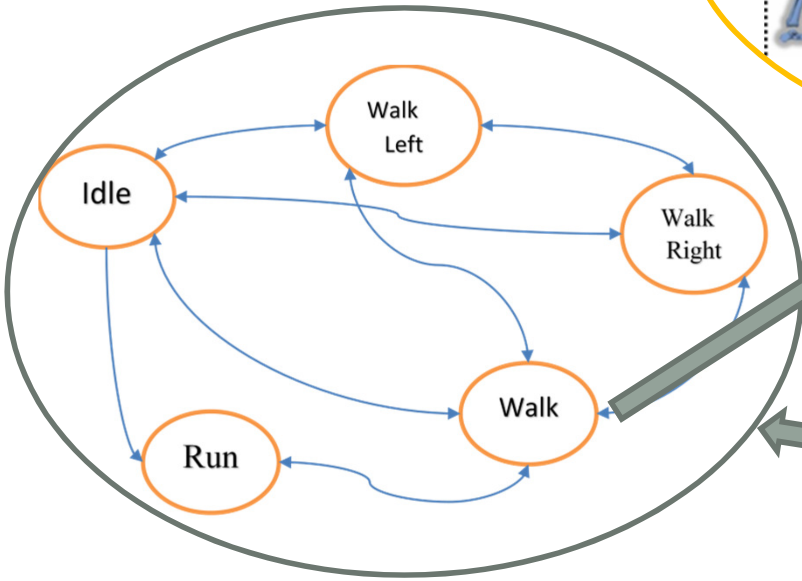


Hirachical FSM

Computer Animation

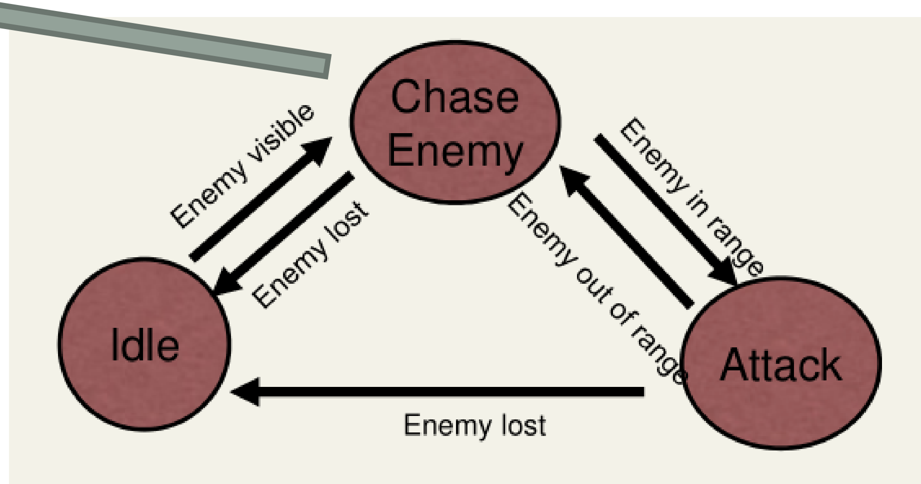


zoom



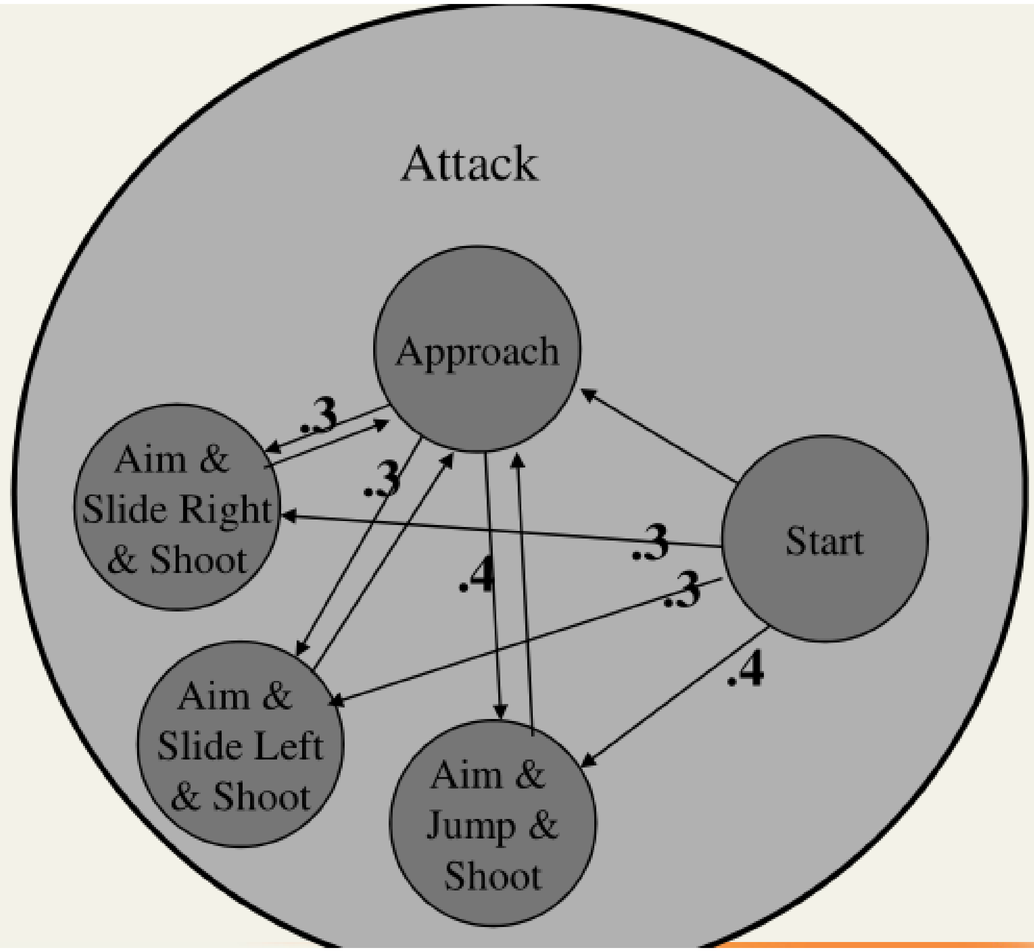
Chase Enemy

zoom



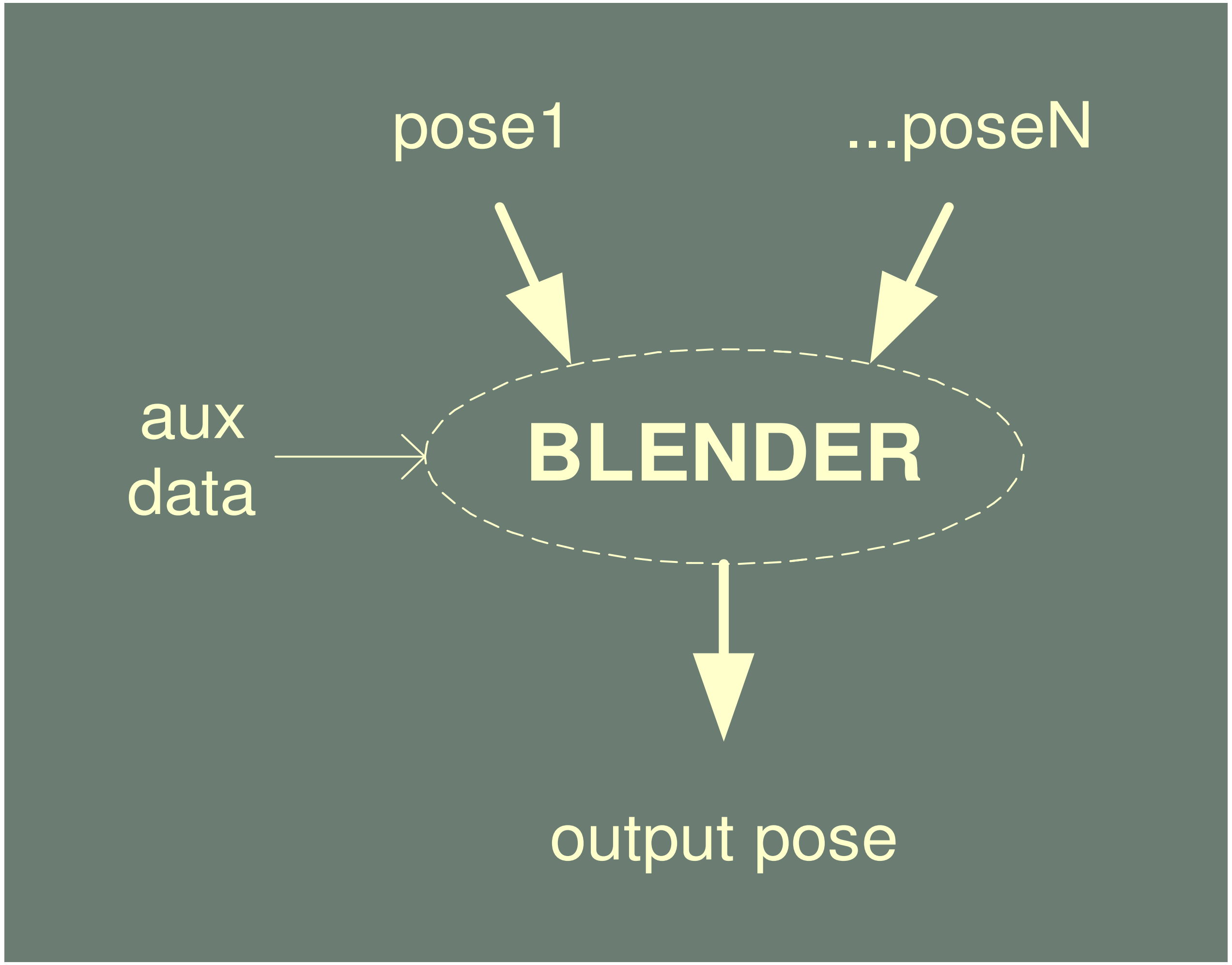
IA

Non Deterministic Hierarchical FSM



- Adds variety to actions
- Have multiple transitions for the same event
- Label each with a probability that it will be taken
- Randomly choose a transition at run-time
- Markov Model: New state only depends on the previous state

Need Generic Blend Operation

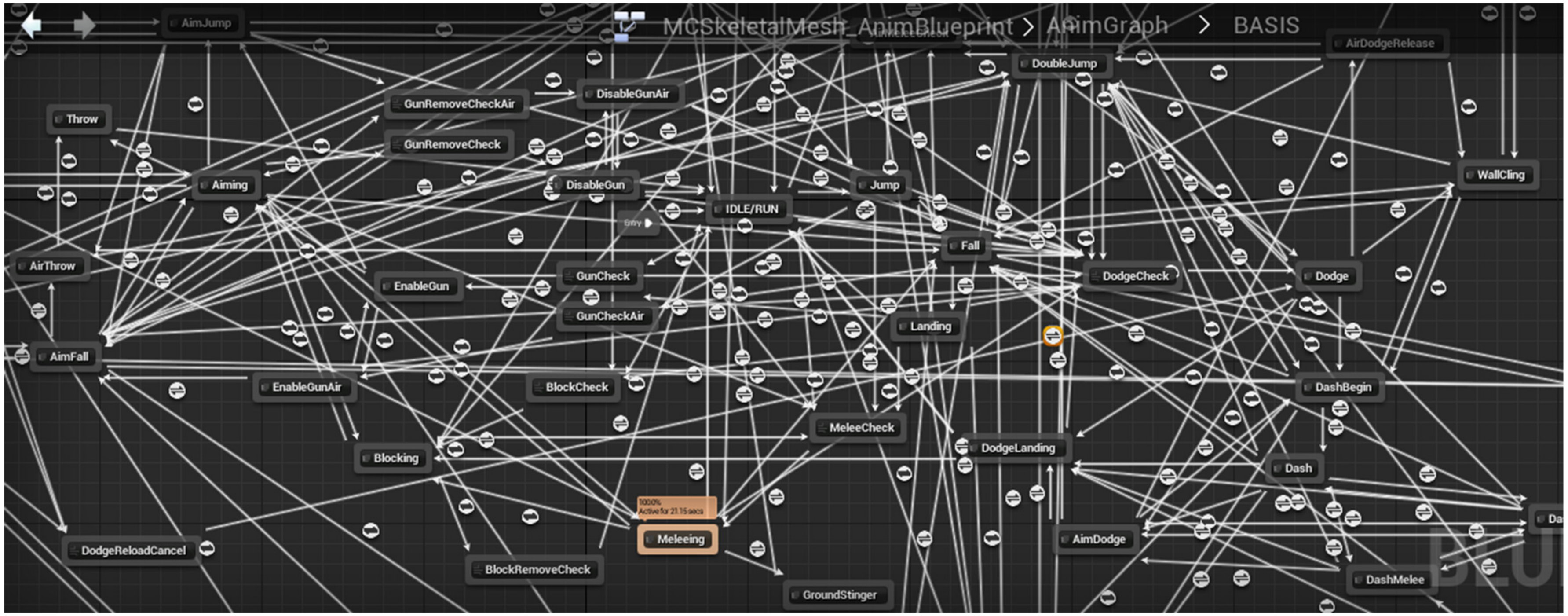


Cf. Quaternion + interpolation

Advantage of FSM ☺

- Very fast – one array access
- Expressive enough for simple behaviors or characters that are intended to be “dumb”
- Can be compiled into compact data structure
  - Dynamic memory: current state
  - Static memory: state diagram – array implementation
- Can create tools so non-programmer can build behavior
- Non-deterministic FSM can make behavior unpredictable

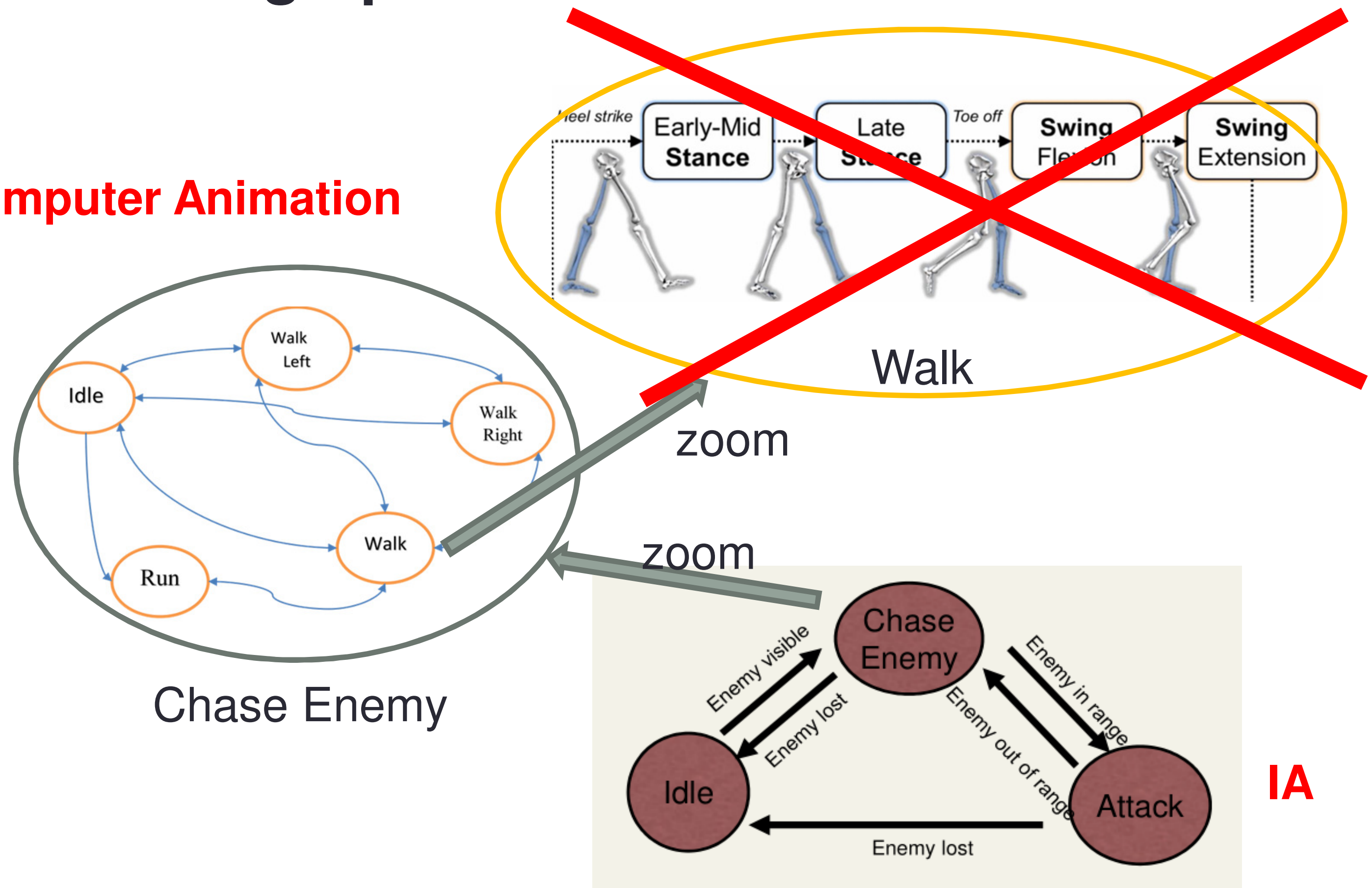
Spaghetti State Machines ☹



Motion Graph [Kovar, Gleicher, Pighin ’02]

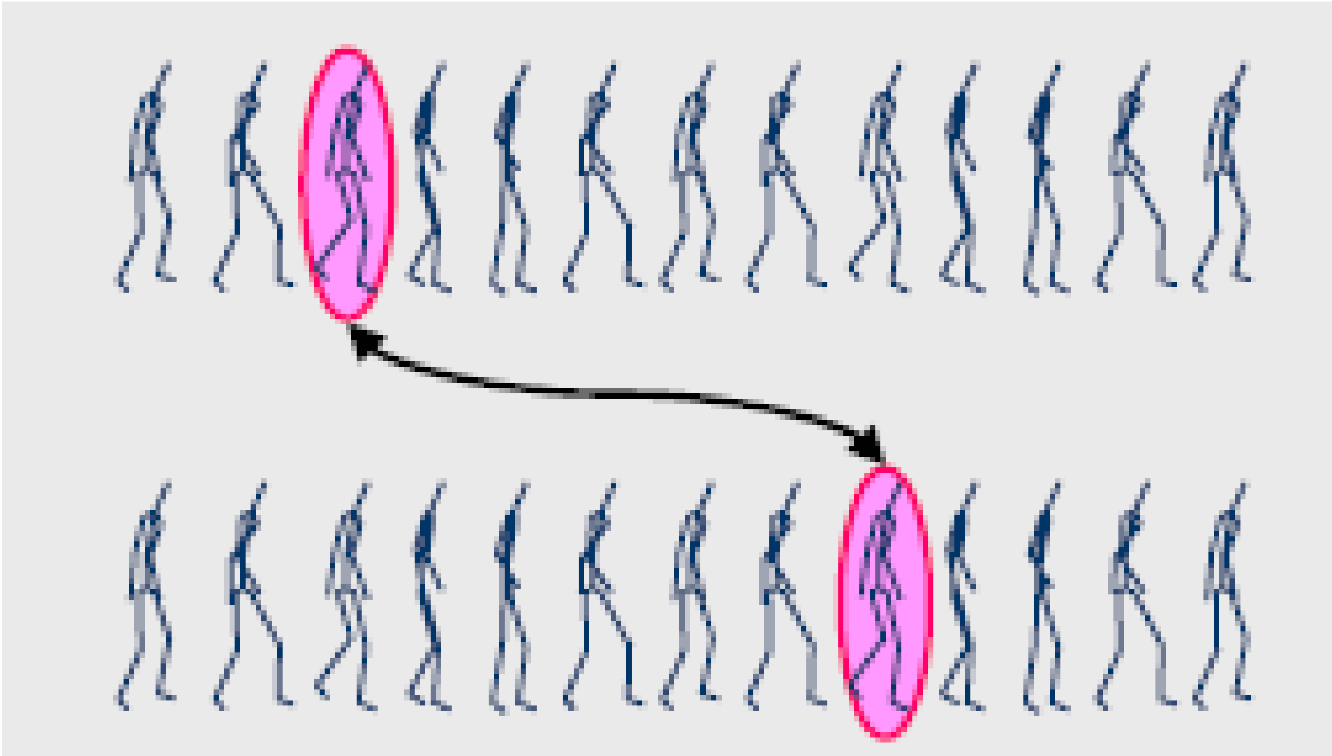
Replace FSM for animation part by an automatically generated graph

Computer Animation



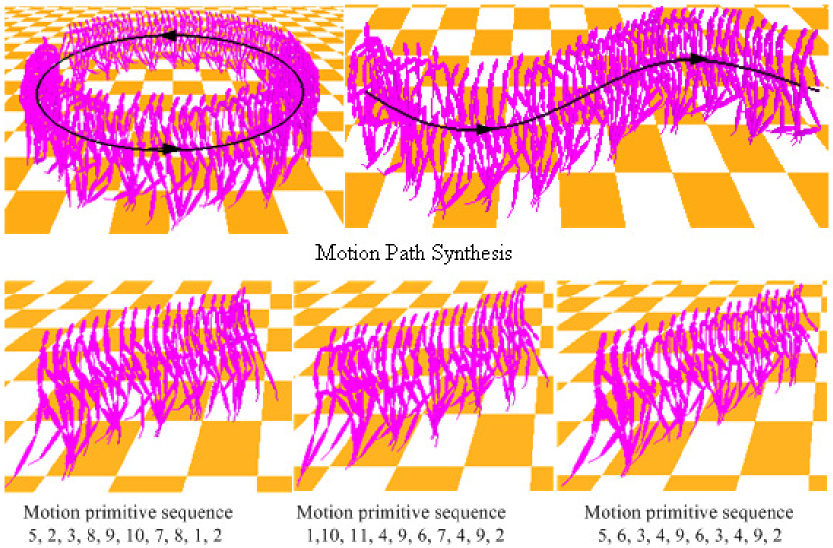
Idea: Motion Graph

Find Automatically Matching States in Motions



Idea: Put Clips Together

- New motions from pieces of old ones!
- Good news:
  - Keeps the qualities of the original (with care)
  - Can create long and novel “streams” (keep putting clips together)
- Challenges:
  - How to decide what clips to connect?
  - How to connect clips?

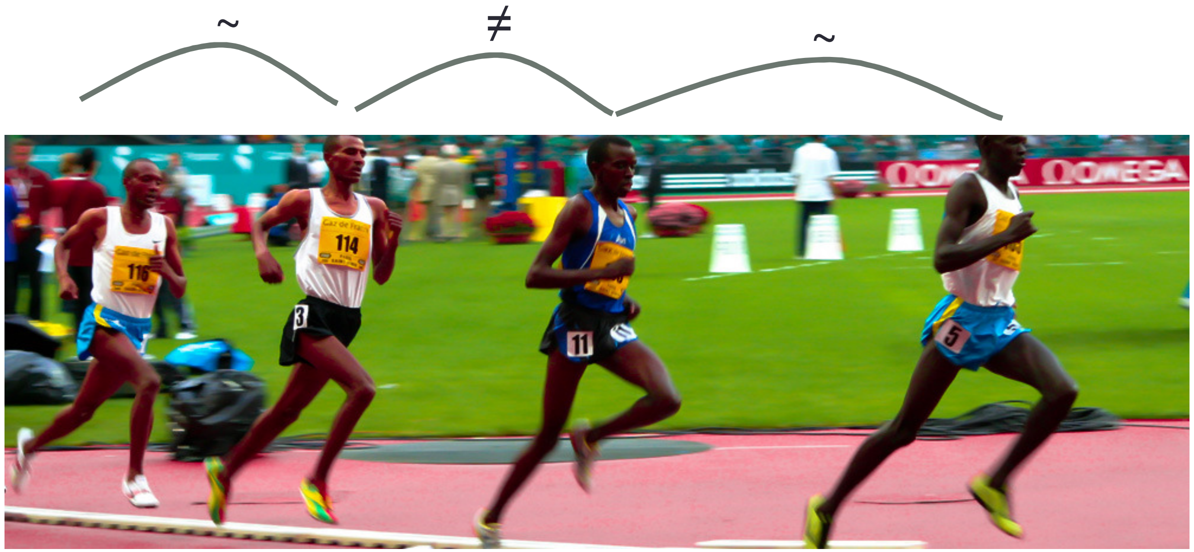


Motion Path Synthesis

Random Sampling for Crowd Simulation

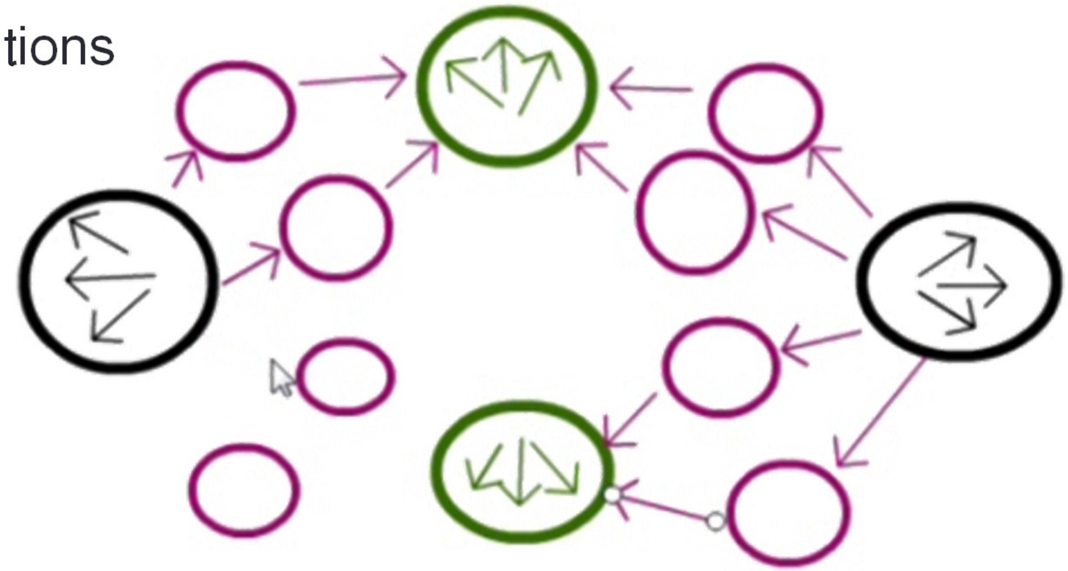
Connecting Clips: Transition Generation

- Transitions between motions can be hard
- Motion interpolation works *sometimes*
  - Blends between aligned motions
  - Cleanup footskate artifacts
- Just need to know when is “sometime”
  - Need a distance between pose



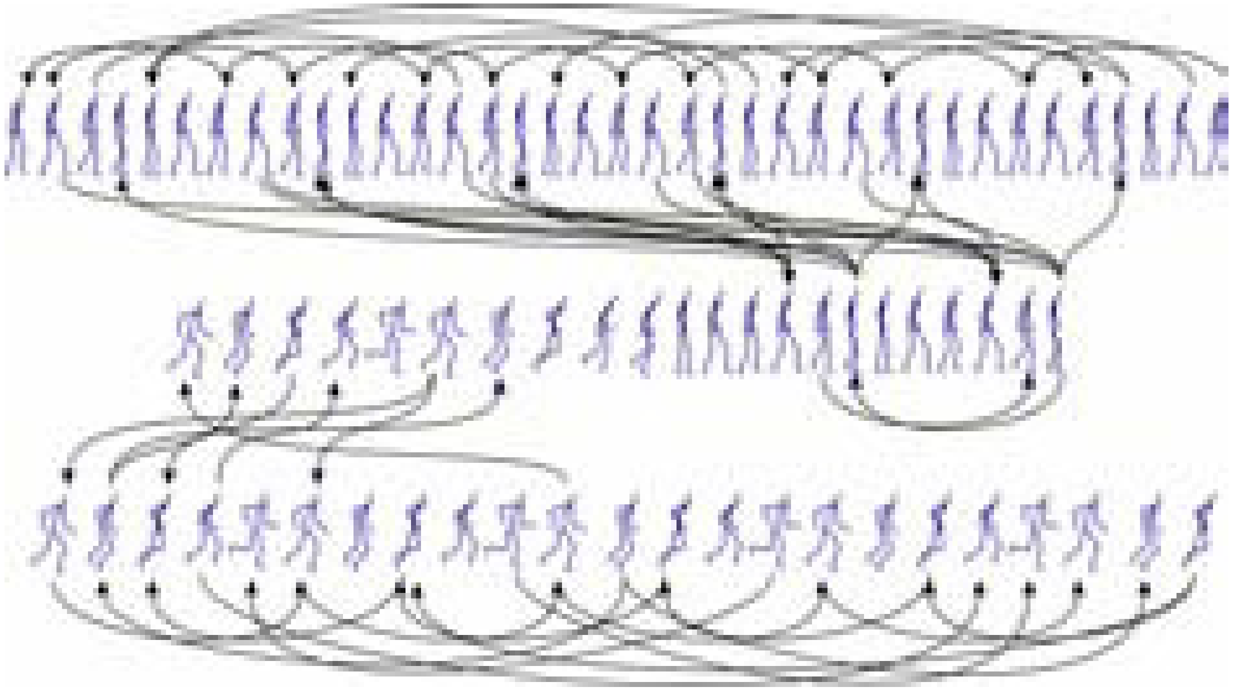
What are motion graphs?

- Directed graph representing a roadmap of motion data for a character
  - Vertex represent a pose in a motion clip
    - Vertex=(motion clip name, pose number)
  - Edges are pose transitions



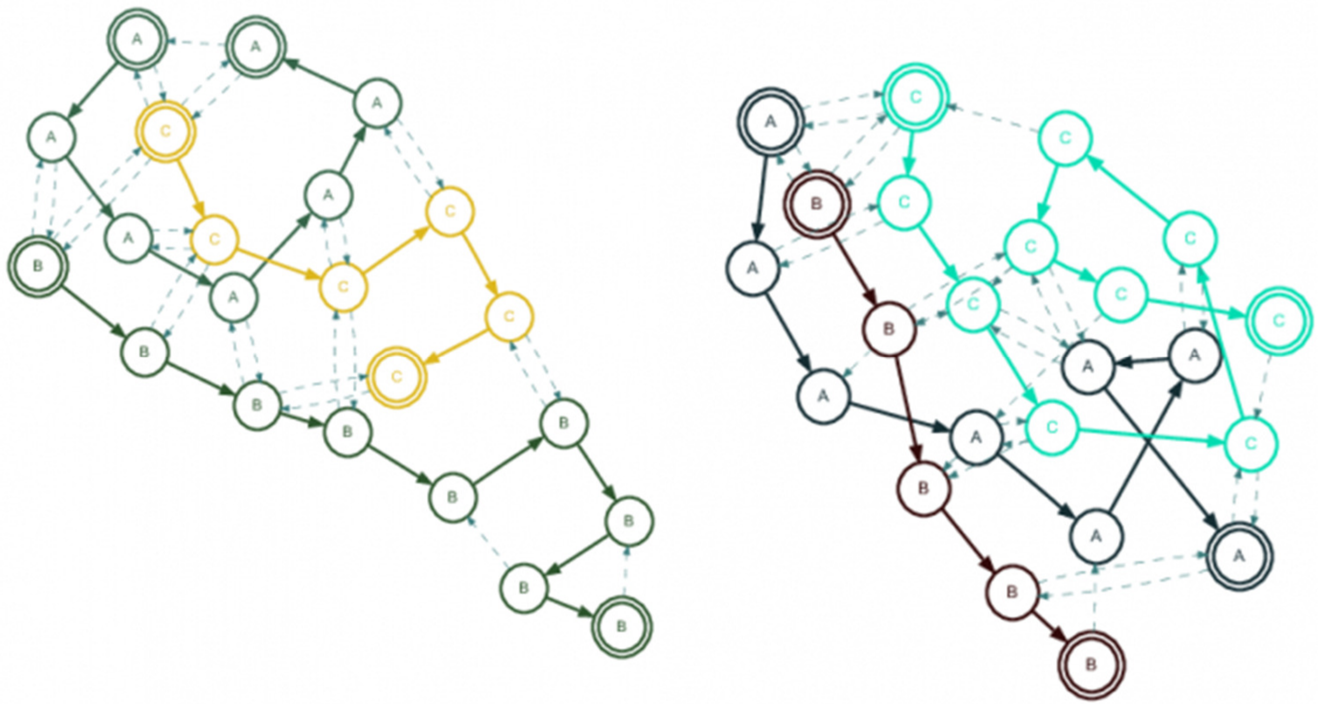
A simple motion graph

- Vertex represent a pose in a motion clip
  =(motion clip name, pose number)
- Edges are pose transitions



A simple motion graph

- Motion Blend & Motion Graph
  - Motion Graph more examples



Building motion graphs

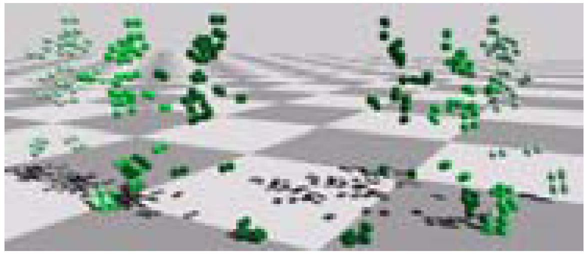
- Identify transition candidates
- Select transition points
- Eliminate problematic edges

Identify transition candidates: pose distance

- For each pose A of clip C_j, calculate its distance to each other pose B of all other clip by basically measuring volume displacement

1) Initial frames we want to compare

2) Extract windows: frame before and after



3) Convert to point clouds

4) Align point clouds and sum squared distances

Identify transition candidates: pose distance

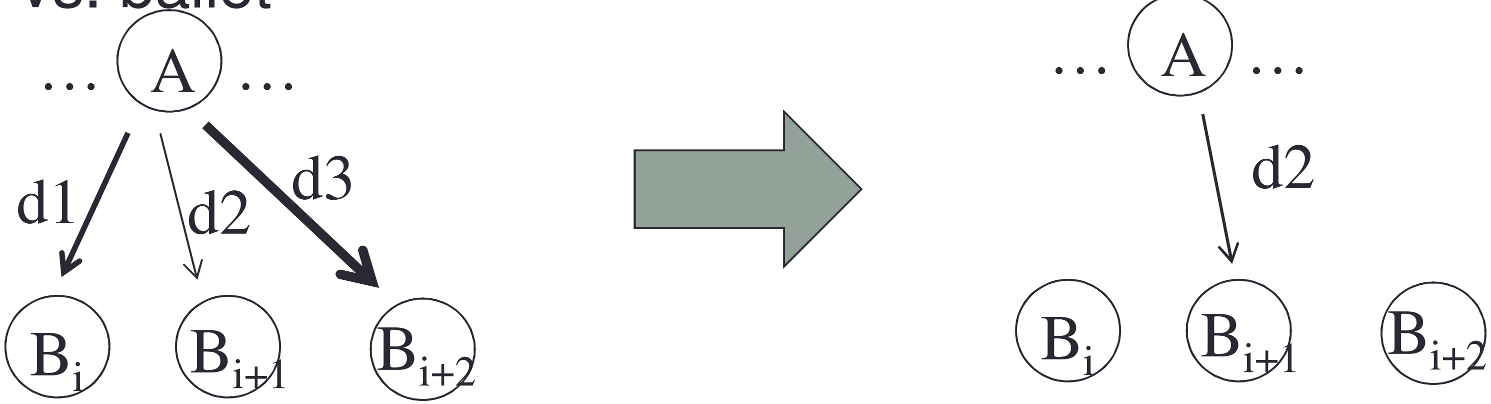
- For each frame/pose A, calculate its distance to each other frame/pose B by basically measuring volume displacement
- Use a weighted point cloud formed over a window of k frames ahead of A and behind B, ideally from the character mesh

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

- Calculate the minimal weighted sum of squared distances between corresponding points, given that a rigid 2D transformation may be applied to the second point cloud

Select transition/edge

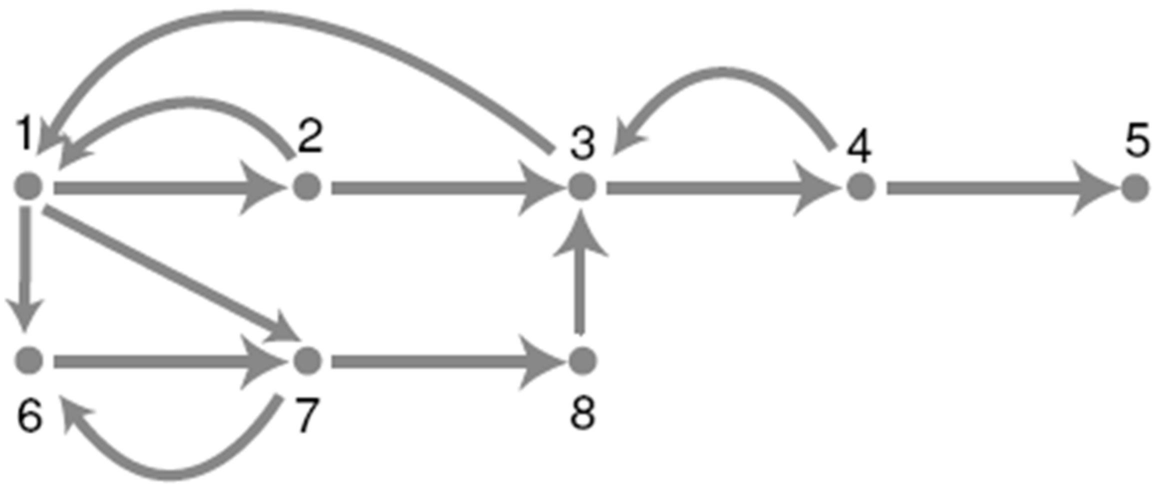
- The previous step gave us all the local minima of the distance function for each pair of points
- Now we simply define a threshold and cut transition candidates with errors above it
- May be done with or without intervention
- Threshold level depends on type of motion – eg. walking vs. ballet



We define transition only between pose with significant similitude

Eliminate problematic edges

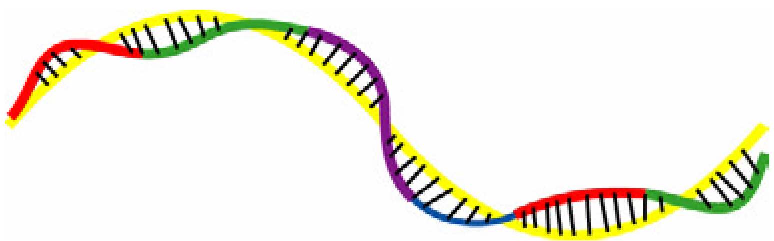
- We want to get rid of:
  - Dead ends – not part of a cycle
  - Sinks – part of one or more cycles but only able to reach a small fraction of the nodes
  - Logical discontinuities – eg. boxing motion forced to transition into ballet motion
- Goal is to be able to generate arbitrarily long streams of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

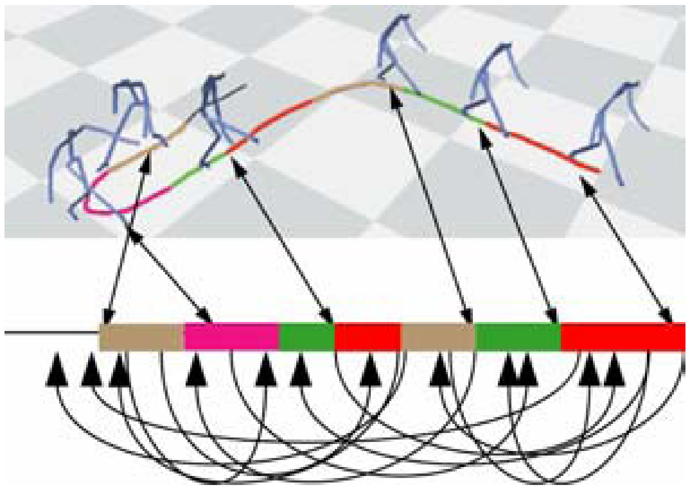


Using a motion graph

- Any walk on the graph is a valid motion
  - Generate walks to meet goals
  - Random walks (screen savers)
  - Search to meet constraints

- Other Motion Graph- like projects elsewhere
  - Differ in details, and attention to detail

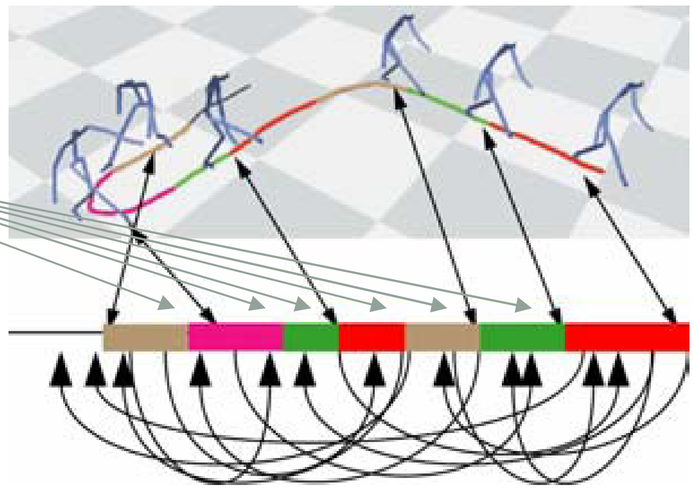




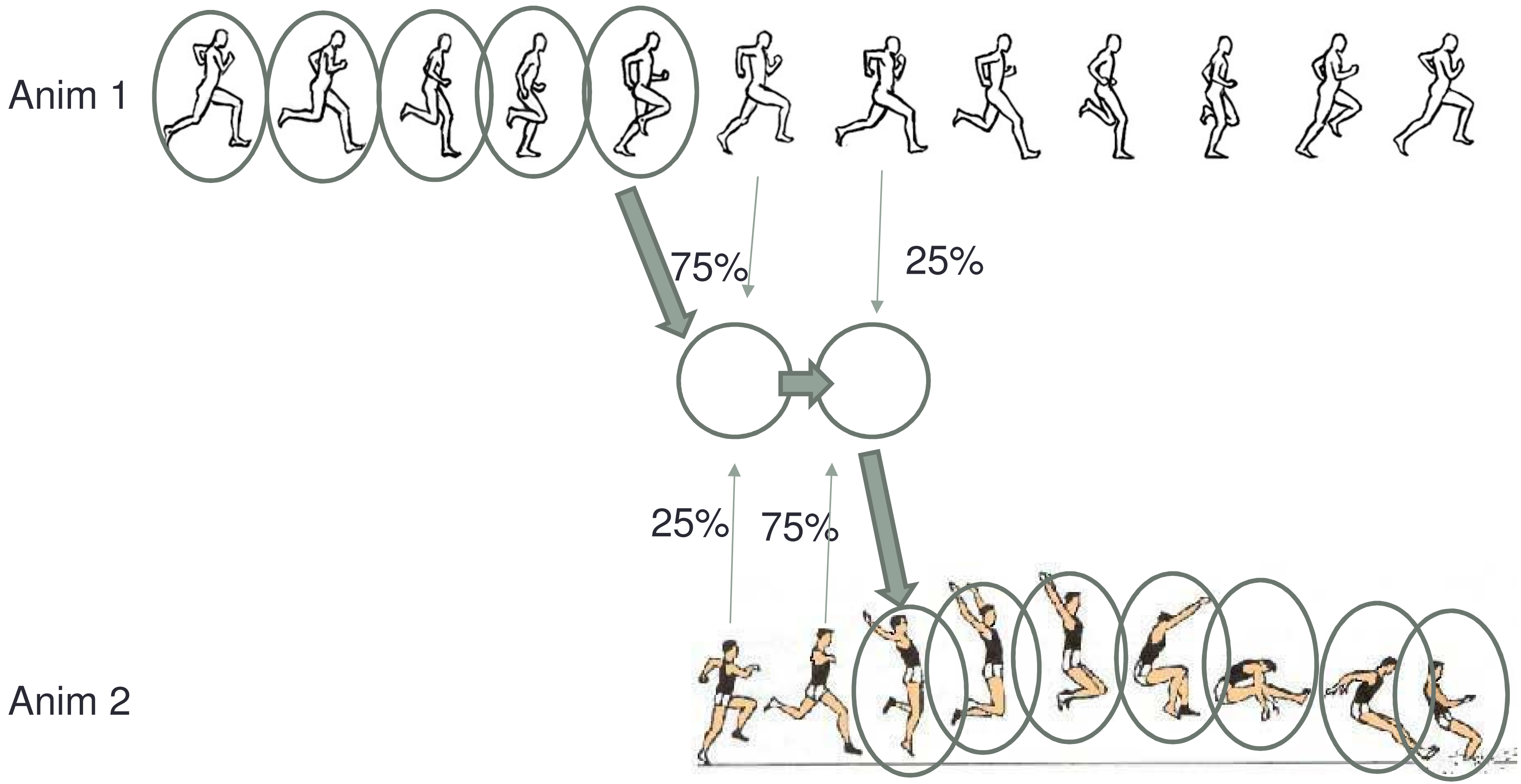
Transitions

- When need to make the transition between frames A_i and B_j blend Ai through A_{i+k-1} with Bj through B_{j-k+1}
  - Align frames with appropriate rigid 2D transformation
  - Use linear interpolation to blend root positions
  - Use spherical linear interpolation to blend joint rotations

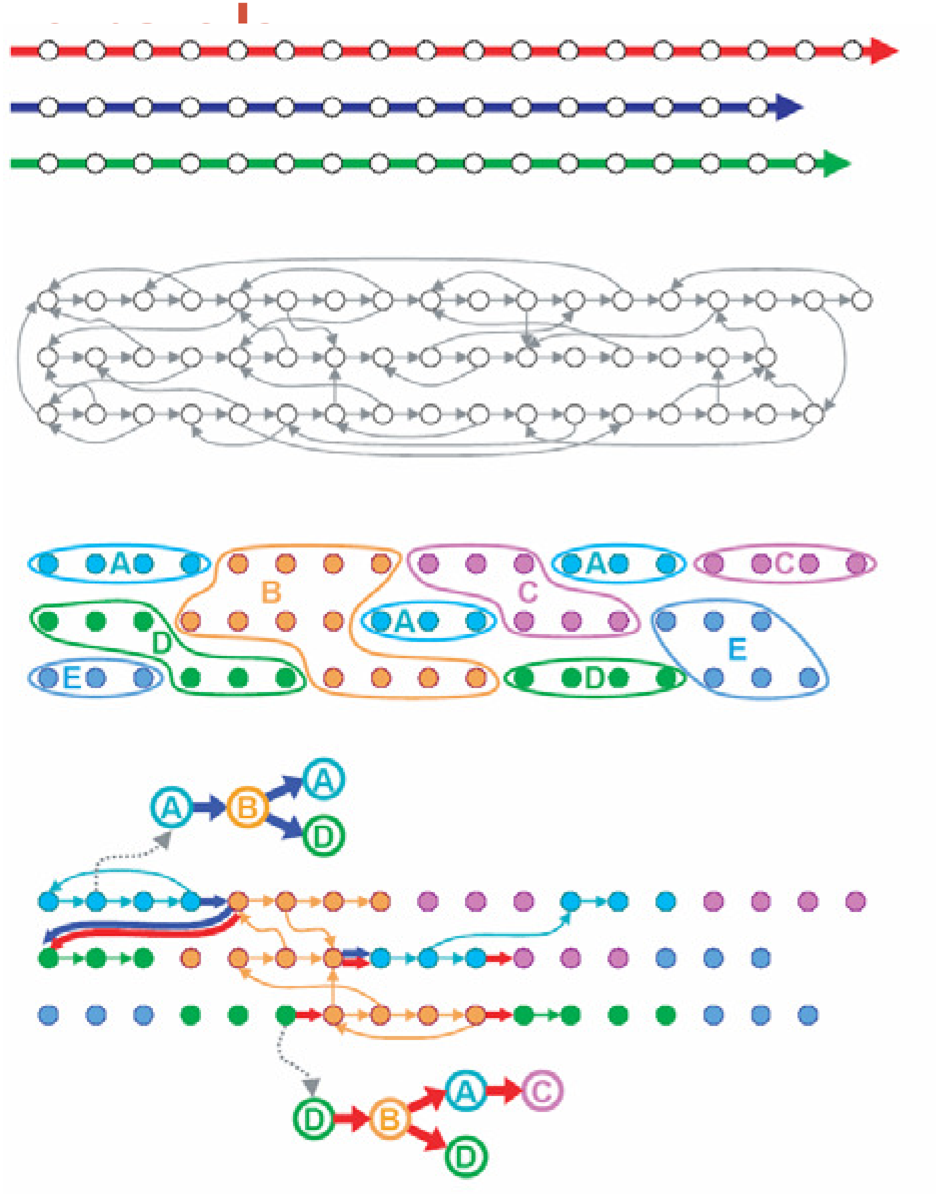
Need transition
Cf. Interpolation



Motion Blending : good transition



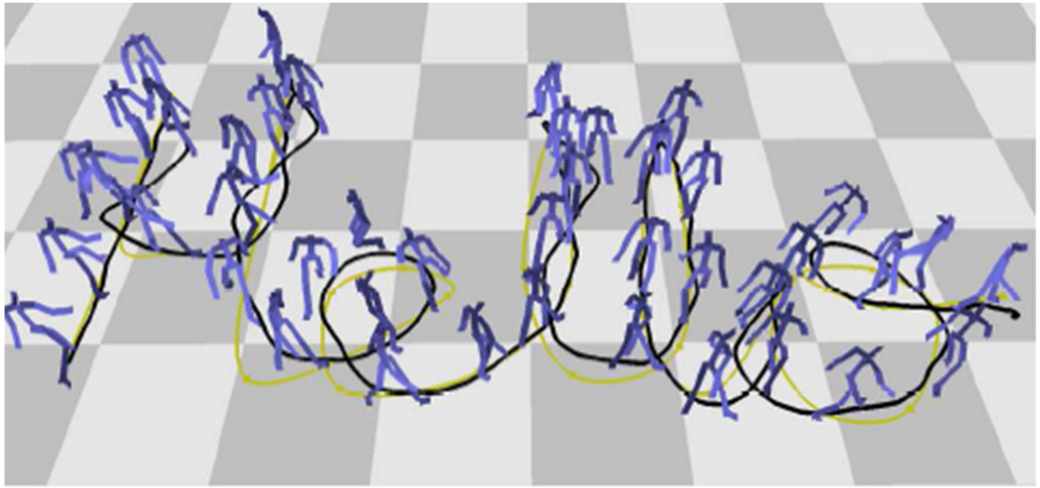
Clustering a motion

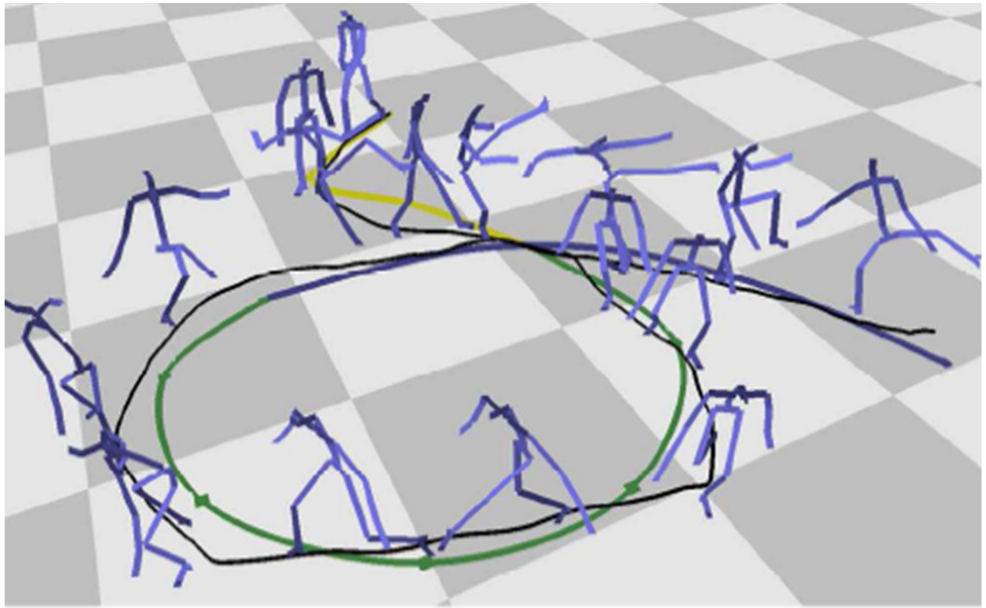


- Clustering the graph
  - For a big graph

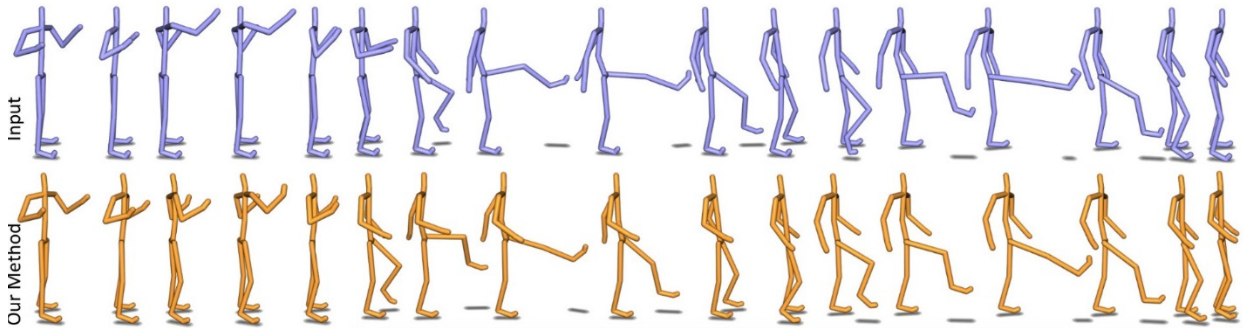
  →Build a meta-graph
  - Improve the exploration

Results





+ video



MOTION EDITING

Input: N animations

Reactivity Problem of Motion Graph

➔Motion Blending between N Animations

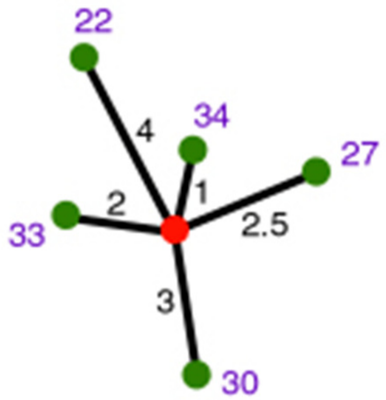
Motion blending

- Often 2 animations not enough to produce realistic moves
- For instance N animations : turn left with different angles
- Interpolating 3 or more angles
  - angle = w_0 x $angle_0$ + w_1 x $angle_1$ + w_2 x $angle_2$
  - avec $\sum w_i = 1$
  - Animations need to be synchronized
- Problem: how to find the weight w_i
  - Inverse distance weighting (See Unity)
  - Barycentric
  - KNN
  - RBF

+VIDEO

Motion blending : barycentric

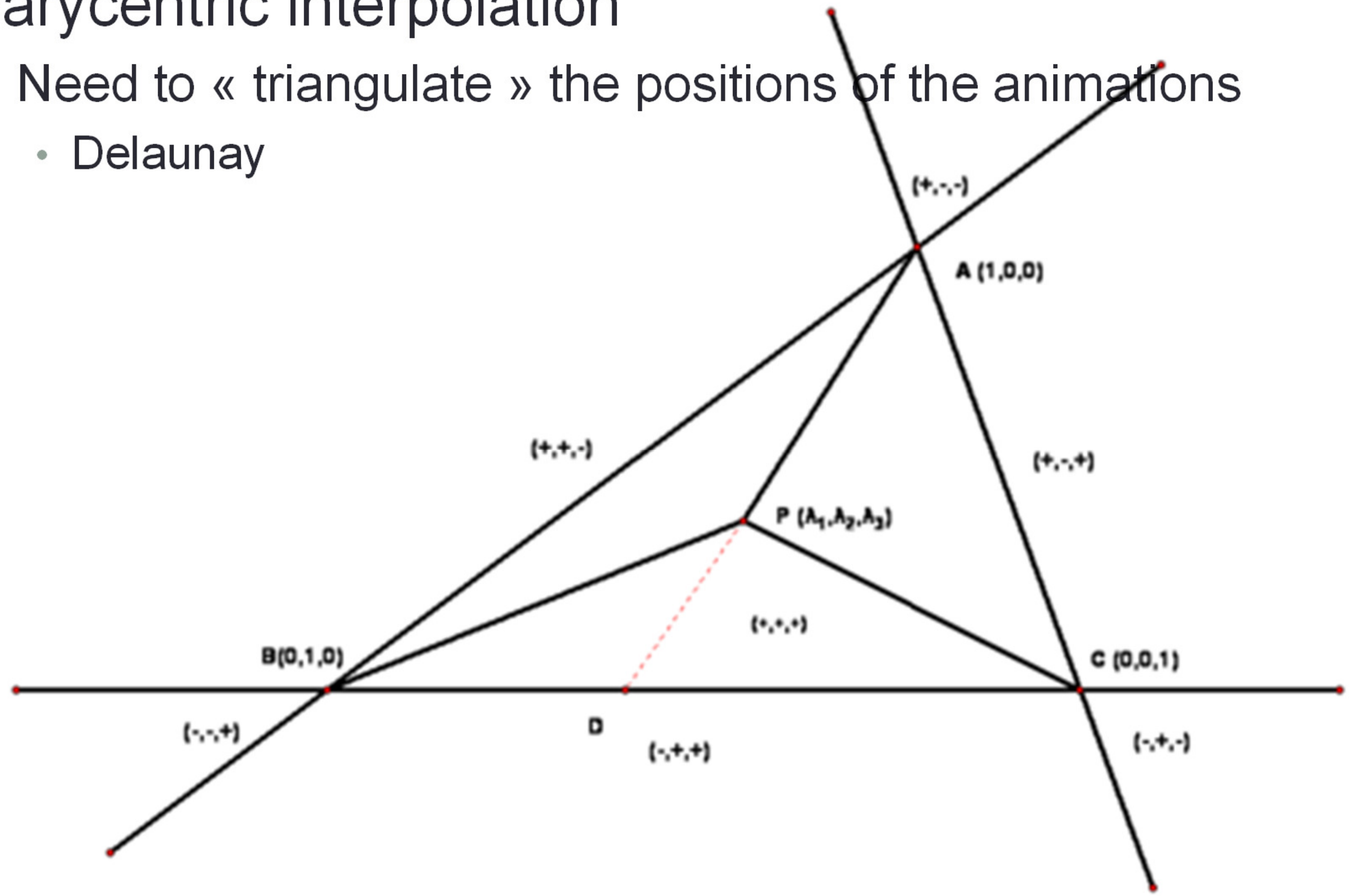
- Inverse distance weighting
  - w_i = 1/distance
  - Normalization of weights
  - Simple computing
- You have to define the position of the animations clips



$$Z(x) = \frac{\Sigma w_i z_i}{\Sigma w_i} = \frac{\frac{34}{1^2} + \frac{33}{2^2} + \frac{27}{2.5^2} + \frac{30}{3^2} + \frac{22}{4^2}}{\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{2.5^2} + \frac{1}{3^2} + \frac{1}{4^2}} = \mathbf{32.38}$$

Motion blending : Barycentric Int

- Barycentric interpolation
  - Need to « triangulate » the positions of the animations
    - Delaunay



Motion blending : RBF

- RBF : Radial Basis Function

$$y(\mathbf{x}) = \sum_{i=1}^{N} w_i \, \phi(\|\mathbf{x} - \mathbf{x}_i\|),$$

- Sum of N radial basis functions, each associated with a different center x_i
- Weight w_i are computed with linear least square method

- Gaussian:

$$\phi(r) = e^{-(\varepsilon r)^2}$$

- Multiquadric:

$$\phi(r) = \sqrt{1 + (\varepsilon r)^2}$$

- Inverse quadratic:

$$\phi(r) = \frac{1}{1 + (\varepsilon r)^2}$$

- Inverse multiquadric:

$$\phi(r) = \frac{1}{\sqrt{1 + (\varepsilon r)^2}}$$

- Polyharmonic spline:

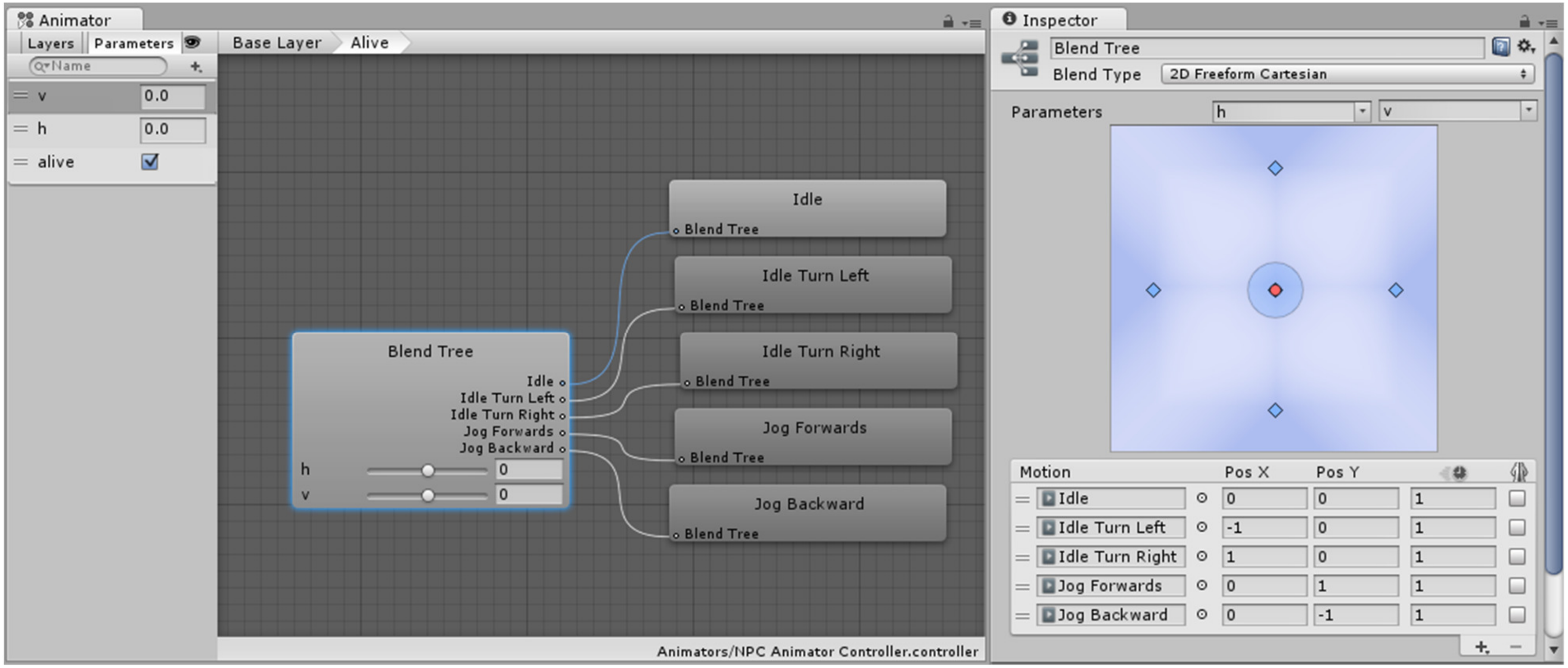
$$\phi(r) = r^k, \qquad k = 1, 3, 5, \dots$$
$$\phi(r) = r^k \ln(r), \qquad k = 2, 4, 6, \dots$$

- Thin plate spline (a special polyharmonic spline):

$$\phi(r) = r^2 \ln(r)$$

Unity : Blend Tree

- Unity : blend tree
  - Finite State Machine
  - Motion blending with inverse distance weighting



Motion Field

- To got further from the interpolation techniques
  - Animation are set on space automatically by a k-nearest neighbor

$$d(m, m') = \sqrt{\begin{array}{rl} \beta_{\text{root}}||v_{\text{root}} - v'_{\text{root}}||^2 & + \\ \beta_0||q_0(\hat{u}) - q'_0(\hat{u})||^2 & + \\ \sum_{i=1}^{n} \beta_i||p_i(\hat{u}) - p'_i(\hat{u})||^2 & + \\ \sum_{i=1}^{n} \beta_i||(q_i p_i)(\hat{u}) - (q'_i p'_i)(\hat{u})||^2 & + \end{array}}$$

- Reinforcement Learning to produce the desired animation
  - States
  - Actions
  - Transition
  - Reward

+VIDEO

Motion Fields for Interactive Character Animation
Lee etal SIGGRAPH 2010

Conclusion

Data-driven motion synthesis

- FSM
- Motion graphs
- Motion blending / Motion interpolation
- Animation control