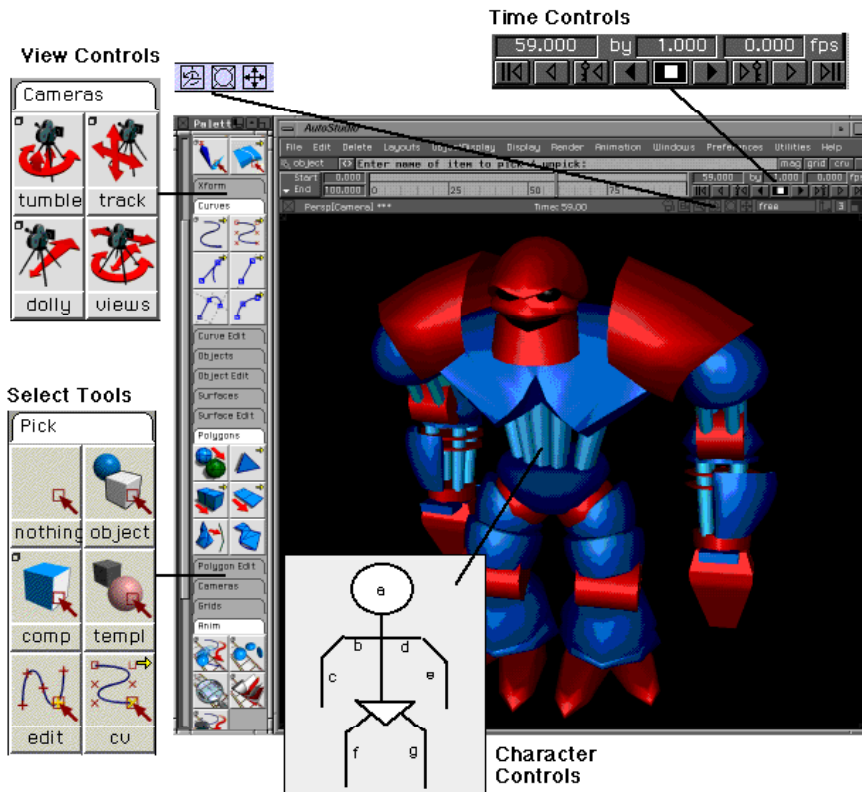


Editing Character Motion

• *Slides from Paul Reitsma*

Motion Data



Keyframe animation



Motion capture

Motion Data

- High quality results
 - Realistic!
 - Beautiful!
 - Perfect!
 - Wrong...
- Need to adapt to changing demands

Motion Editing

- Getting the motion you *want* from the motion you *have*
- General techniques
 - Warping
 - Splicing
 - Blending
 - Transplanting

Warping Motions

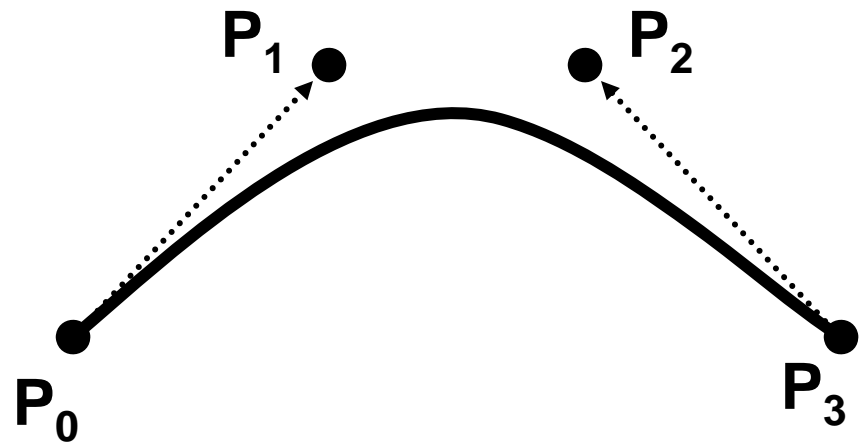
- Adapt a single motion
 - e.g., character needs to catch a ball
 - A catching motion is available
 - Problem: that motion catches where ball *isn't*
- Common solution:
 - Inverse kinematics
 - *Displacement splines*

Displacement Splines

- Spline: smooth piecewise polynomial curve
- Displacement: distance away
- *Displacement spline*: smooth curve giving distance away from the original motion data
 - Smoothly moves animation away from original data to target point

Displacement Splines

- e.g., cubic Bezier curve
 - four control points P_i
 - $B(t) = (1-t)^3P_0 + 3t(1-t)^2P_1 + 3t^2(1-t)P_2 + t^3P_3$
 - $B(0) = P_0$
 - $B(1) = P_3$
 - $B'(0) = 3(P_1 - P_0)$
 - $B'(1) = 3(P_3 - P_2)$

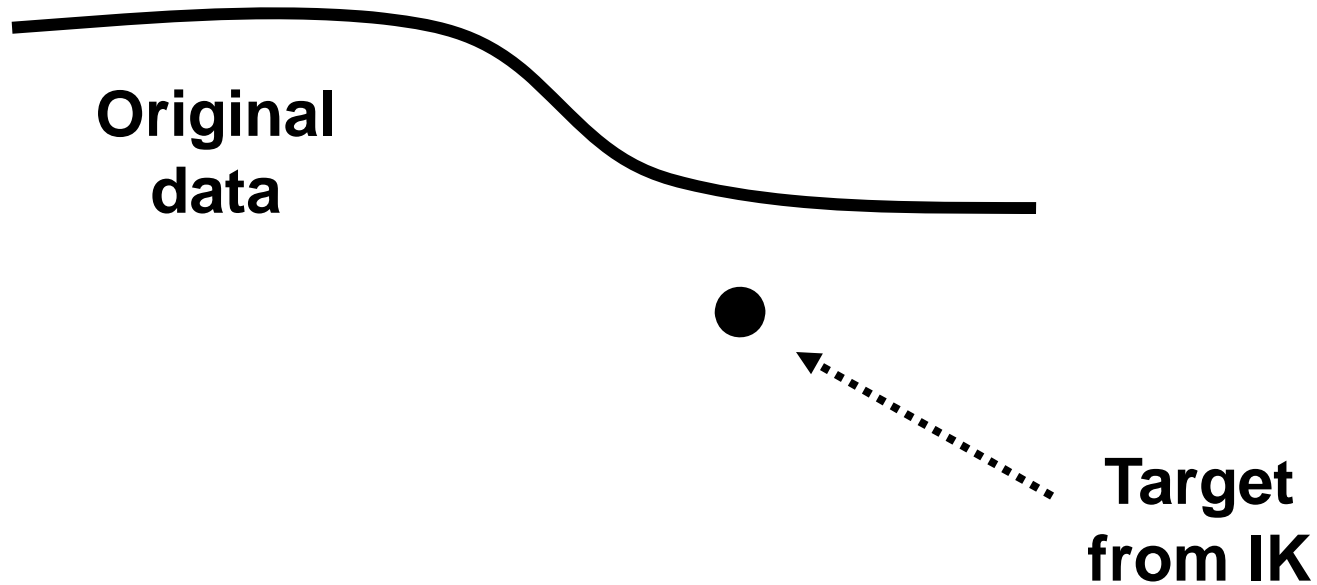


Displacement Splines

- e.g., cubic Bezier curve
 - Create longer curve via sequential cubics
 - Overlap first point of B_{i+1} with last point of B_i to make (C0) continuous curve
 - Overlap two points to make smooth C1 (velocity) continuous curve
 - Smoother curves require more advanced splines (e.g., cubic uniform B-splines)

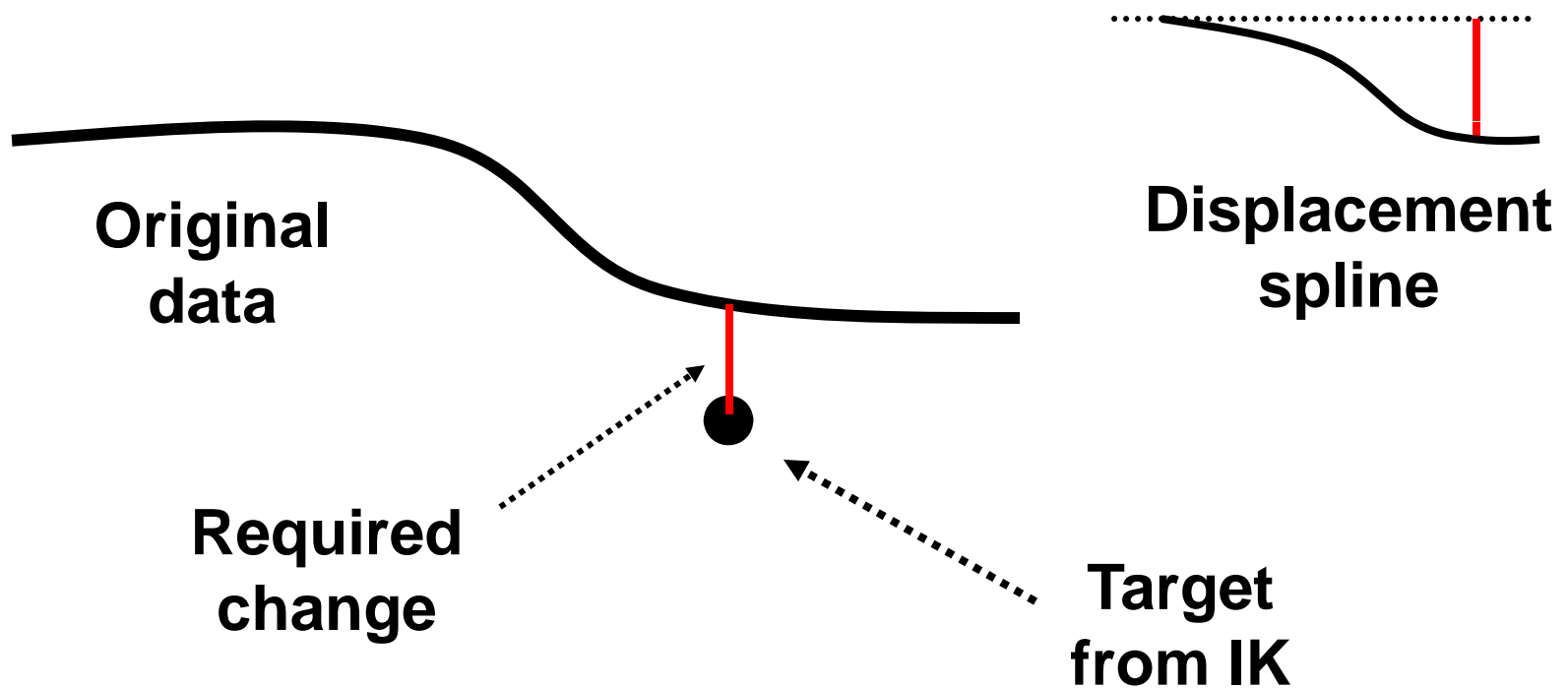
Warping Motions

- e.g., angle of elbow joint



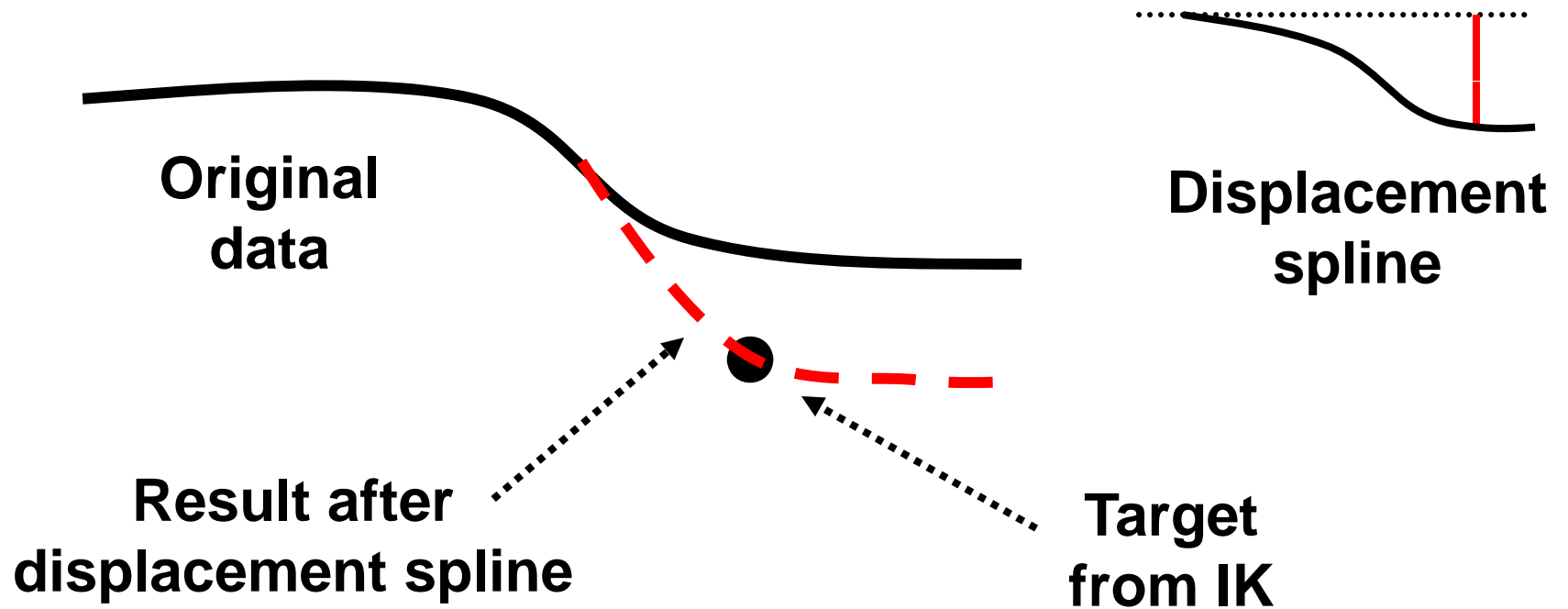
Warping Motions

- e.g., angle of elbow joint



Warping Motions

- e.g., angle of elbow joint



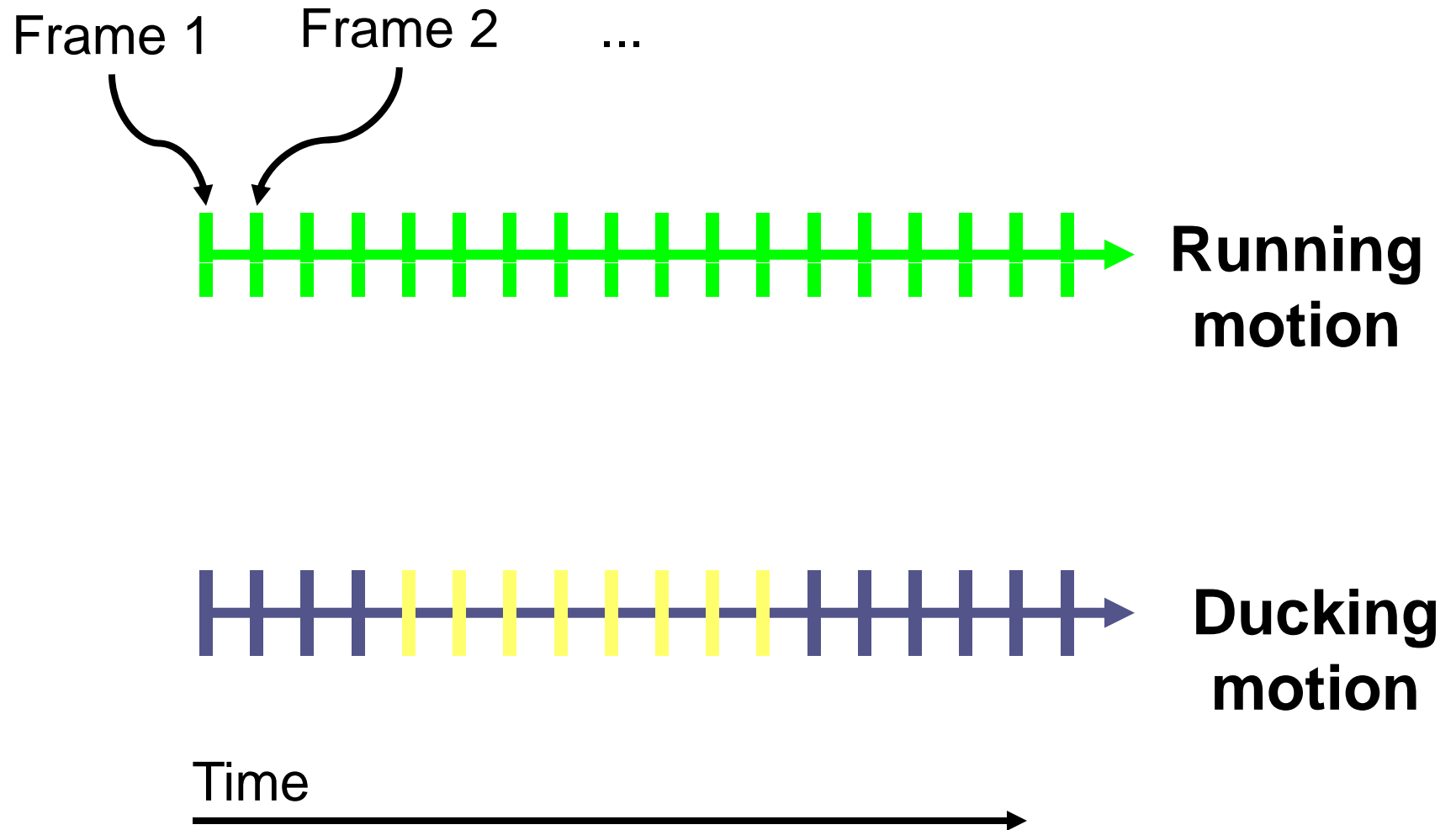
Warping Motions

- Also used to adjust locomotion
 - Longer steps
 - Higher jump
 - Proper turning angle
 - Duck under a beam
- General approach for adjusting or adding constraints to a single motion

Splicing Motions

- Have running motion
- Have ducking motion
- Want a motion where the character runs and then ducks

Example: Run + Duck



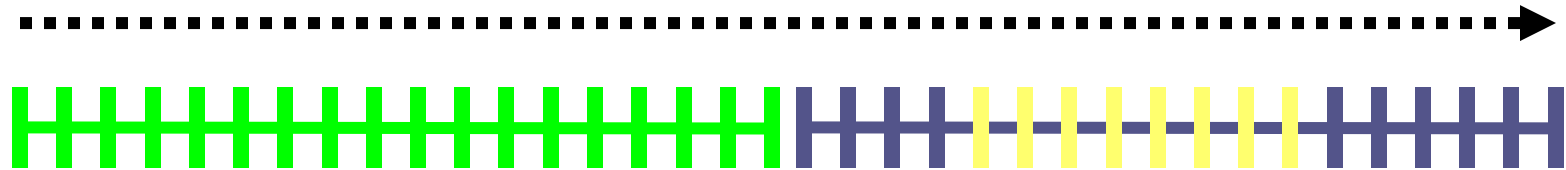
Approaches

1. Play running then ducking clips

Approaches

1. Play running then ducking clips

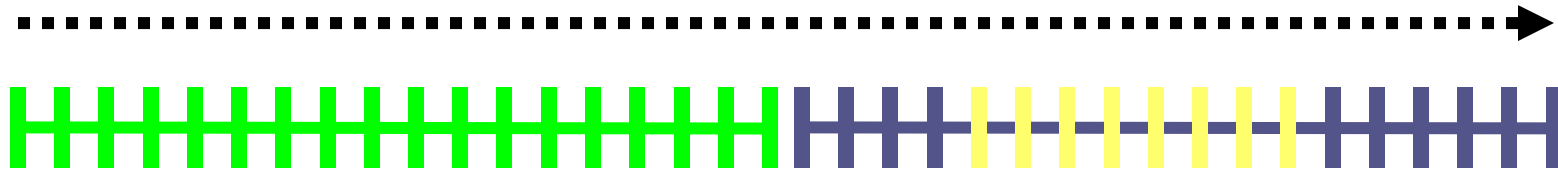
Time



Approaches

1. Play running then ducking clips

Time



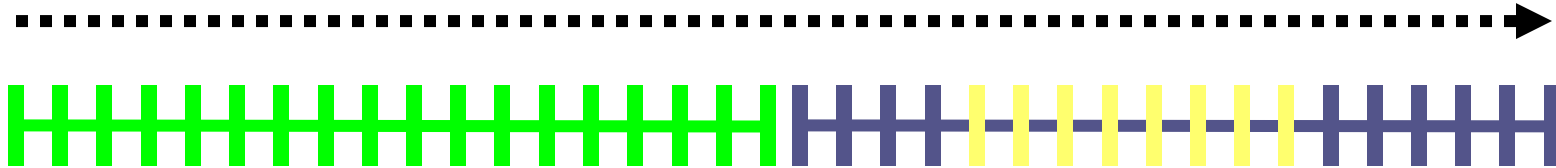
Time until character has ducked



Approaches

1. Play running then ducking clips

Time



Time until character has ducked



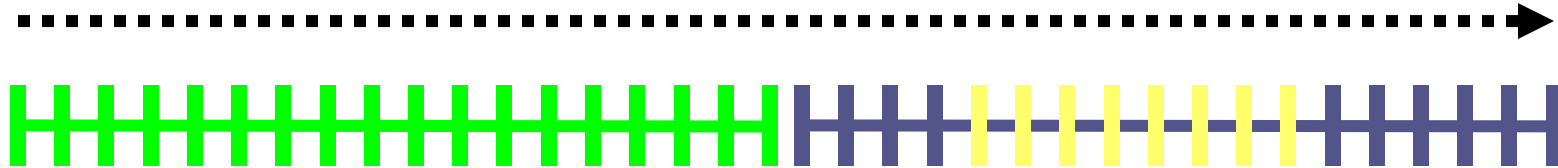
Time until boulder hits character's head



Approaches

1. Play running then ducking clips

Time



Time until character has ducked



Time until boulder hits character's head



Time until lose interest in game



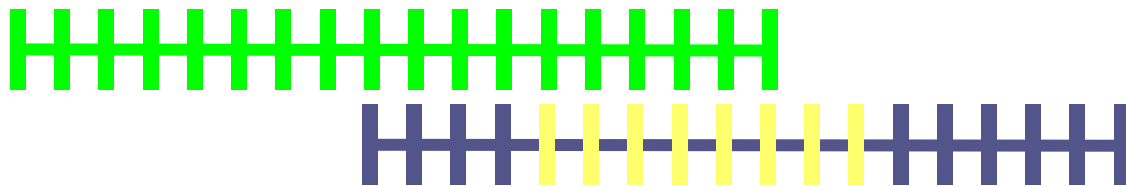
Approaches

1. Play running then ducking clips
 - Problem: no control over timing
2. Cut from running clip to ducking when needed

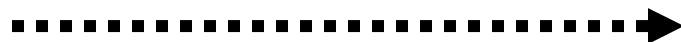
Approaches

1. Play running then ducking clips

Time



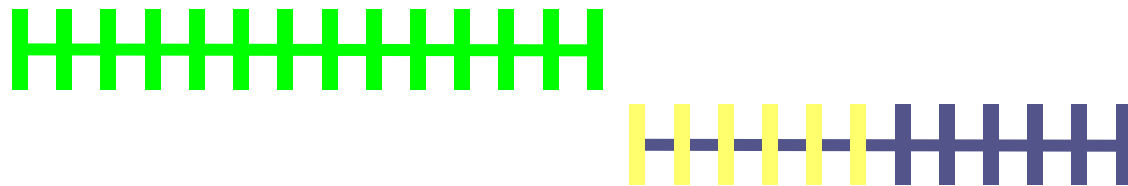
Time until boulder hits character's head



Approaches

1. Play running then ducking clips

Time



Time until boulder hits character's head



Approaches

1. Play running then ducking clips

Time



Time until character has ducked



Time until boulder hits character's head



Approaches

1. Play running then ducking clips

Time



Time until character has ducked



Height of character's head



Approaches

1. Play running then ducking clips
2. Cut from running clip to ducking when needed
 - Problem: bad transitions

Approaches

1. Play running then ducking clips
2. Cut from running clip to ducking when needed
 - Problem: bad transitions
 - *Aesthetic* problem (user preference)

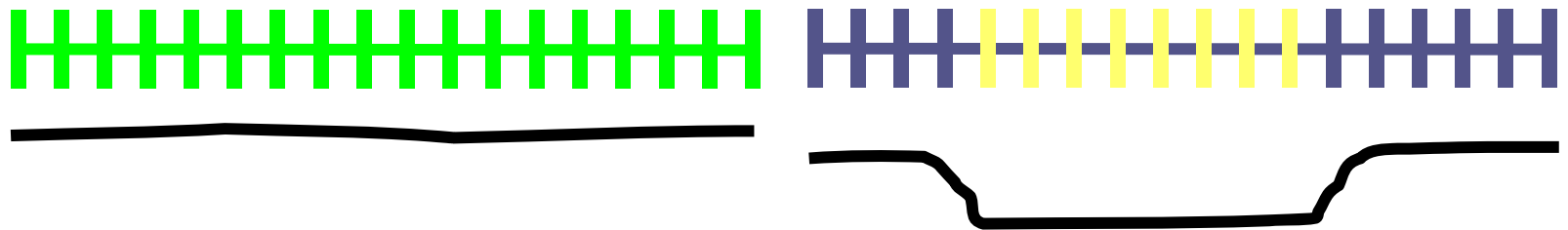
Avoiding Bad Transitions

1. Capture the aesthetics

■ *Motion similarity metric*

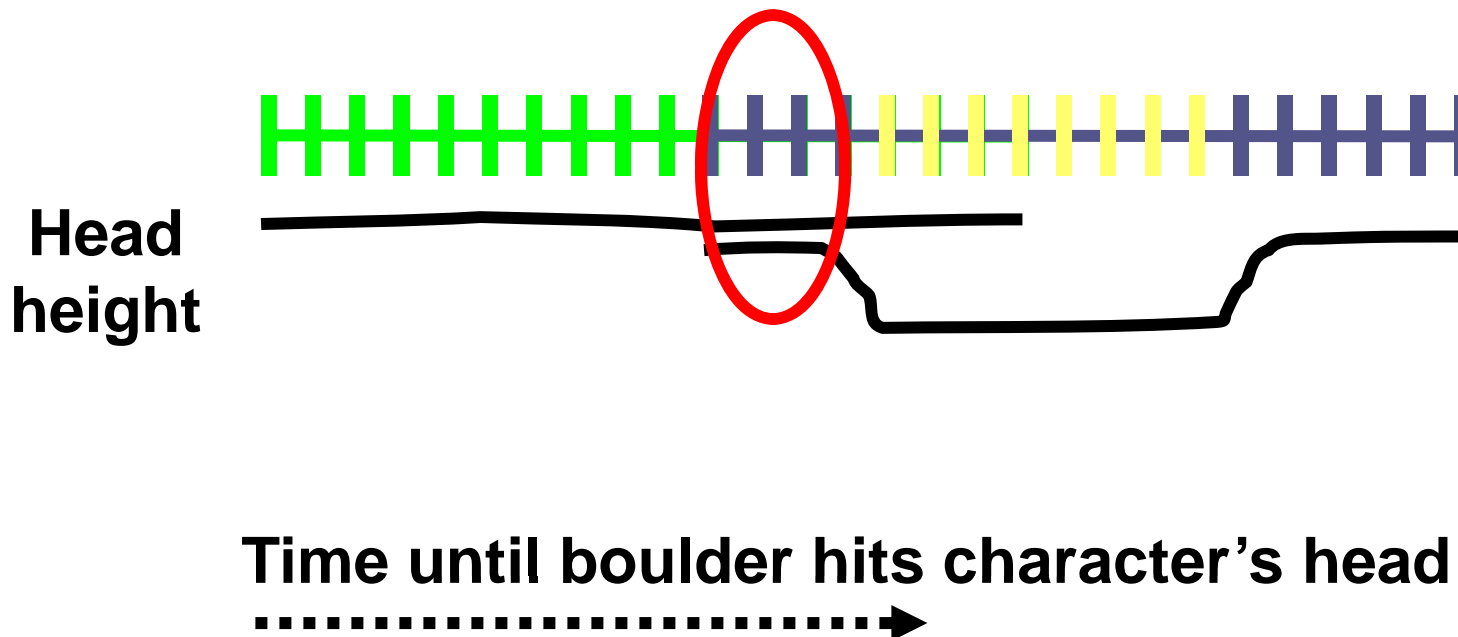
■ Combine motions at highly similar points

Head
height



Avoiding Bad Transitions

1. Capture the aesthetics
 - *Motion similarity metric*
 - Combine motions at highly similar points



Avoiding Bad Transitions

1. Capture the aesthetics
 - *Motion similarity metric*
 - Combine motions at highly similar points
2. Smooth remaining gaps
 - *Motion smoothing algorithm*

Motion Similarity Metric

- Where are motions similar enough to move from one to the other with minimal disruption?
 - Pose similarity
 - Velocity similarity
 - Interaction similarity

Pose Similarity

- Simplest: joint angle differences
 - Relative importance (shoulder vs. wrist)
 - e.g. $\sum w(j) * (P(j_s) - P(j_t))^2$
- Alternatives
 - Alignment of virtual points on body
 - Visual difference of projection
 - Perceptual model

Velocity Similarity

- To avoid splicing a touch and a punch
- Options:
 - Compute numerical derivative
 - $V(t) = P(t) - P(t-1)$
 - Match poses over time window

Interaction Similarity

- Contact with the environment should remain crisp
 - Smoothing often loses this
 - e.g. feet sliding on/in ground
- Pose/velocity measures enforce this poorly
- May want to penalize score if motions differ in interaction with environment
 - e.g., setting foot down vs. standing still

Motion Smoothing Algorithm

- Three basic choices

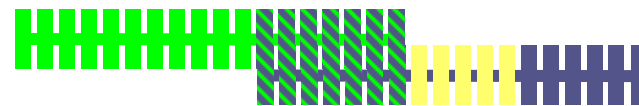
1. Connecting



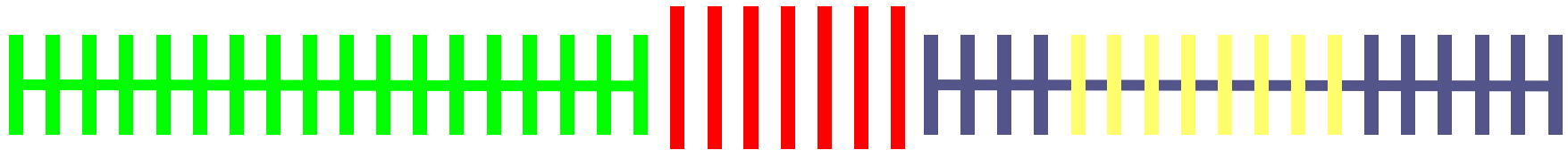
2. Warping



3. Blending



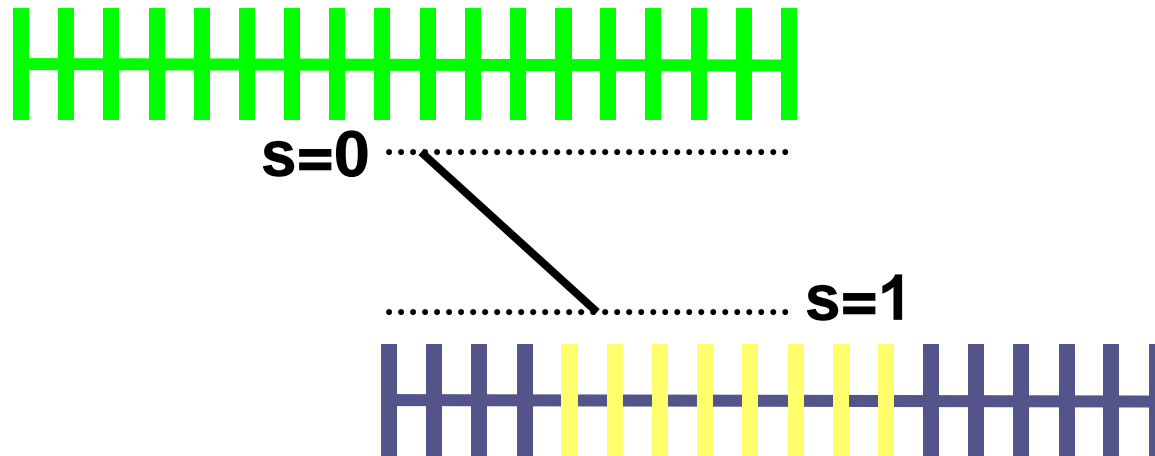
Connecting Motions



- Synthesize frames to fill the gap
 - Typically interpolate between poses
 - e.g., SLERP
 - Physics-based optimization also possible
 - Minimum-energy trajectory between poses

Blending Motions

- Blend smoothly from motion A to motion B
 - Need blending algorithm, blending weights
 - $P(t) = (1-s)P_A(t) + sP_B(t), 0 \leq s \leq 1$



Blending Motions

- Not only for transitioning
 - e.g. Have a running motion
 - Have a walking motion
 - Want a *jogging* motion

Blending Motions

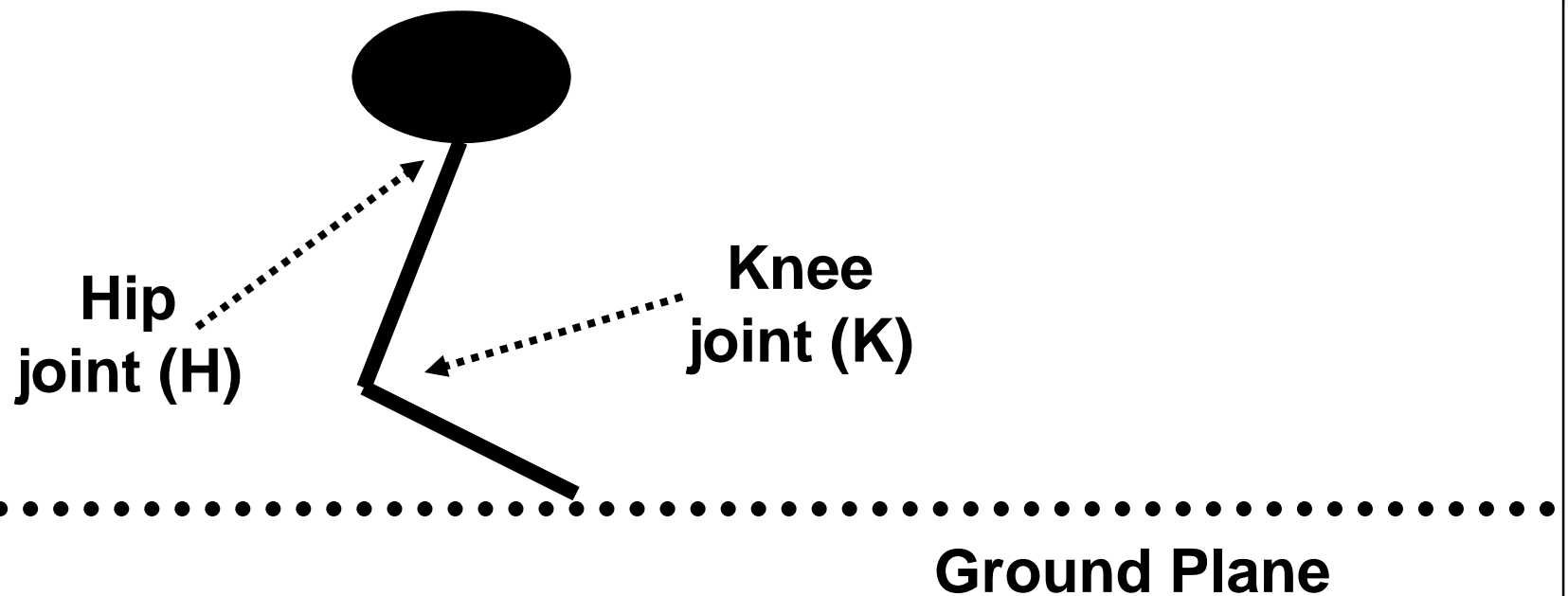
1. Blend the joint angles

Blending Motions

1. Blend the joint angles
 - Problem: cosine is nonlinear

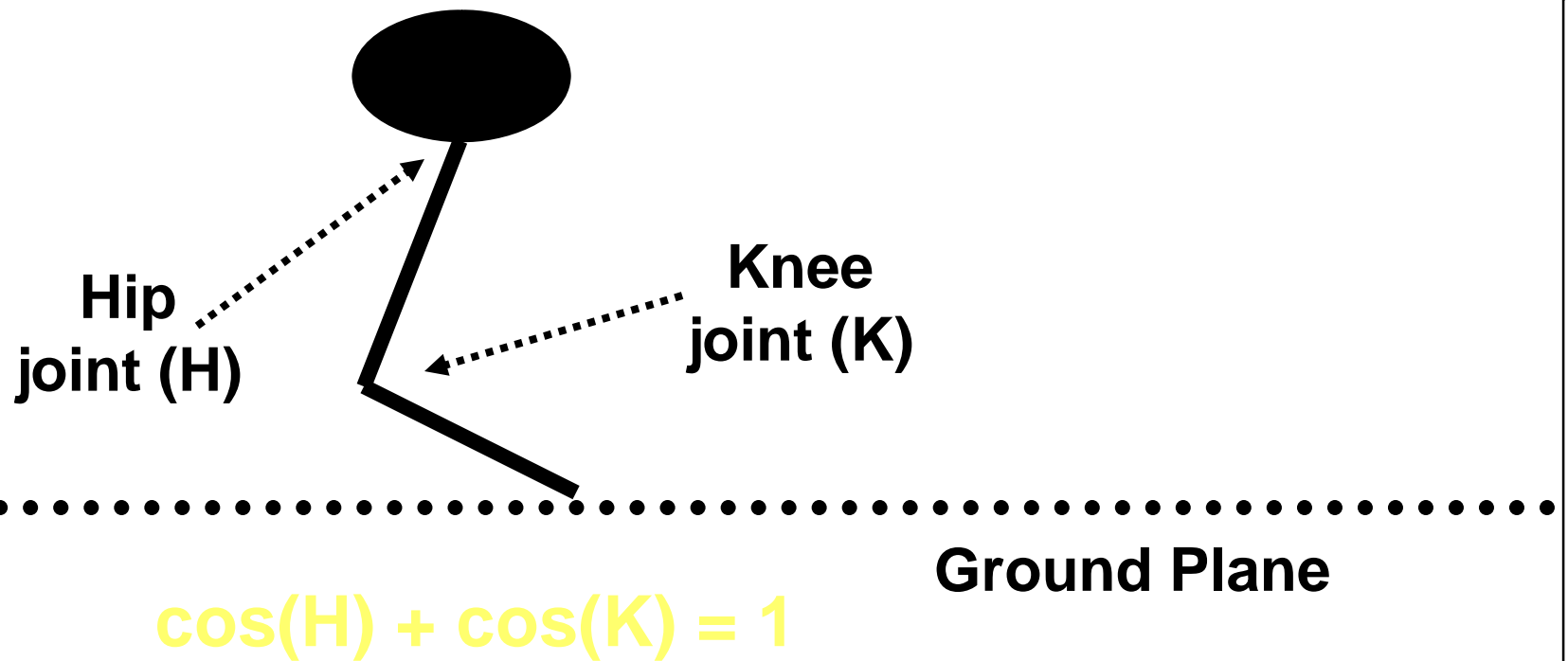
Blending Motions

1. Blend the joint angles
 - Problem: cosine is nonlinear



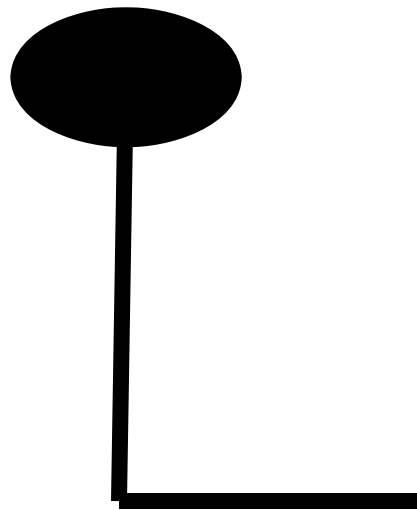
Blending Motions

1. Blend the joint angles
 - Problem: cosine is nonlinear



Blending Motions

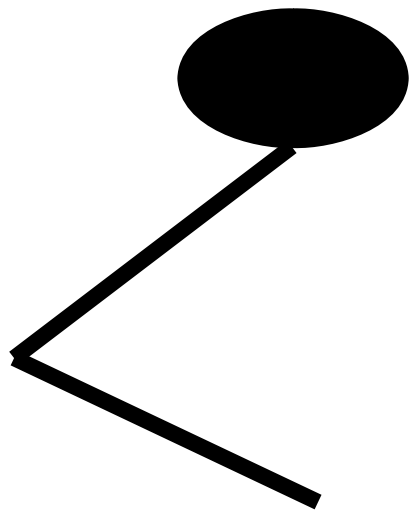
1. Blend the joint angles
 - Problem: cosine is nonlinear



$$\cos(0) + \cos(90) = 1$$

Blending Motions

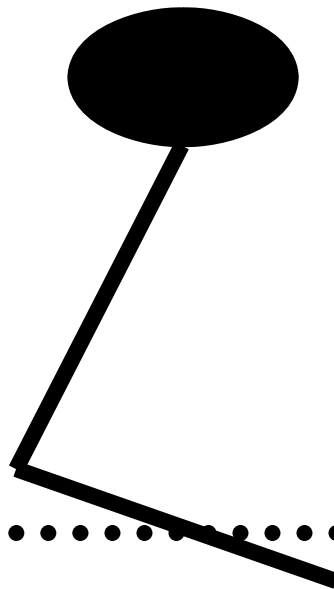
1. Blend the joint angles
 - Problem: cosine is nonlinear



$$\cos(50) + \cos(70) = 1$$

Blending Motions

1. Blend the joint angles
 - Problem: cosine is nonlinear



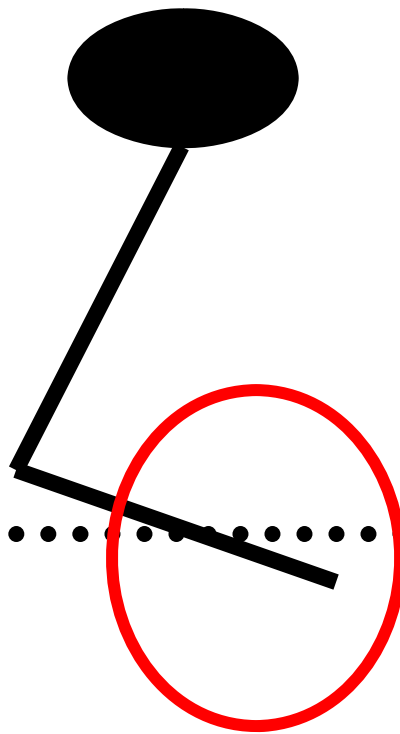
$$\cos(0) + \cos(90) = 1$$

$$\cos(50) + \cos(70) = 1$$

$$\cos(25) + \cos(80) = 1.1$$

Blending Motions

1. Blend the joint angles
 - Problem: cosine is nonlinear



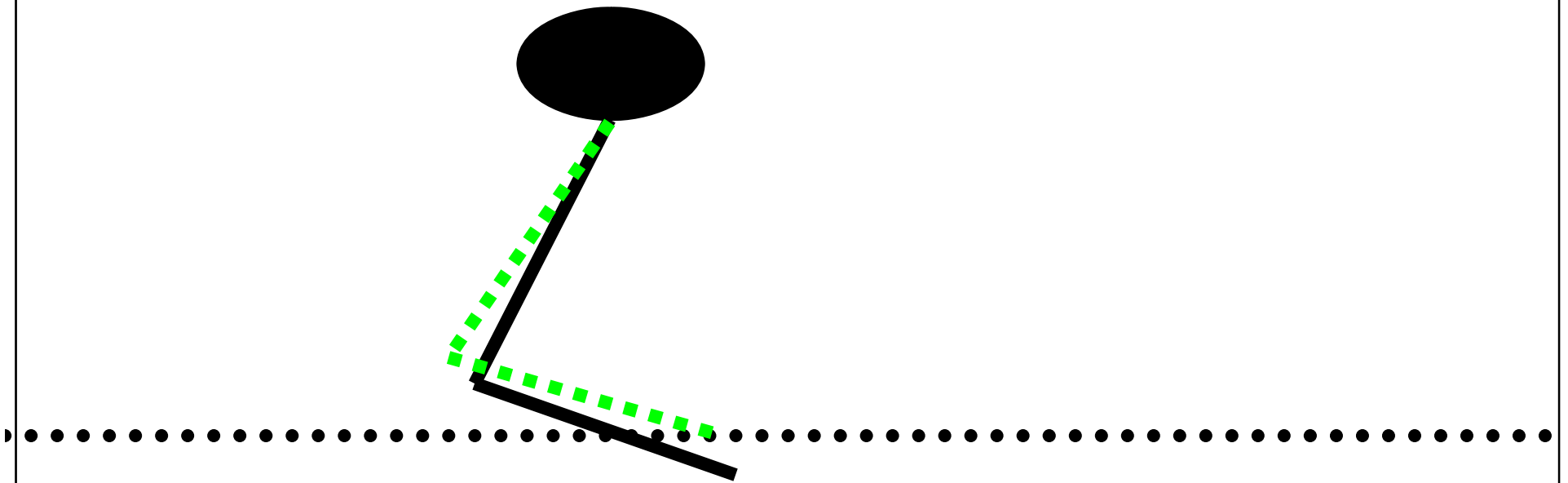
**Breaks interactions
with environment**

Blending Motions

1. Blend the joint angles
 - Problem: cosine is nonlinear
 - Breaks interactions with environment
 - Solution: *IK and motion warping*

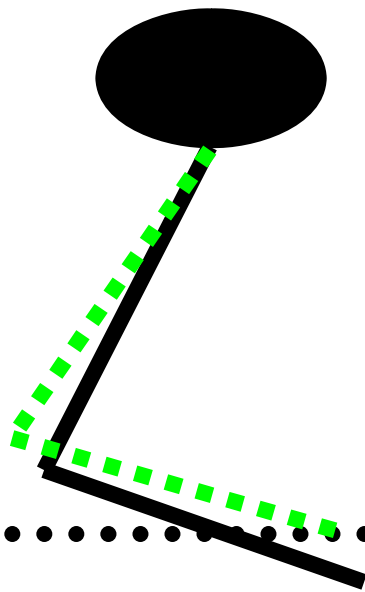
Blending Motions

1. Blend the joint angles
 - Post-processing fix with IK



Blending Motions

1. Blend the joint angles
 - Post-processing fix with IK

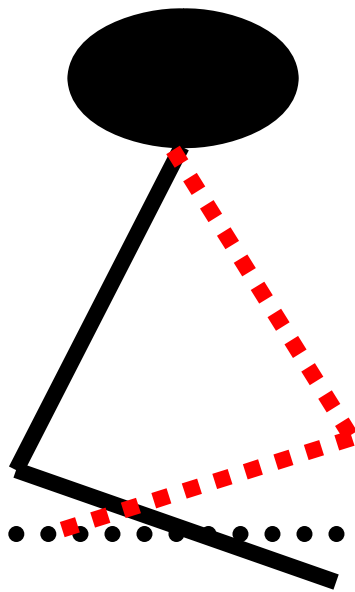


IK tricks:

- joint limits
- natural poses
- solution coherency

Blending Motions

1. Blend the joint angles
 - Post-processing fix with IK

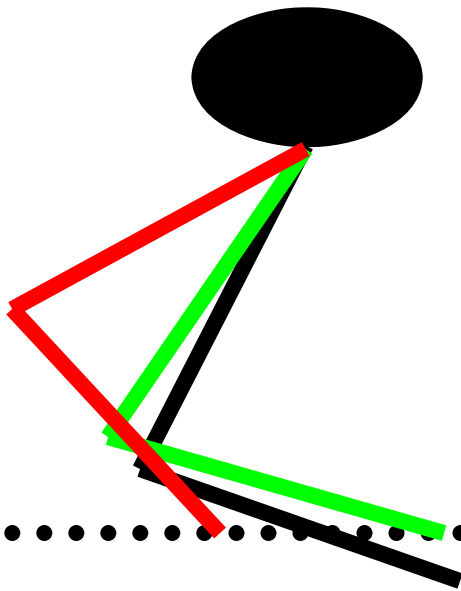


IK tricks:

- joint limits
- natural poses
- solution coherency

Blending Motions

1. Blend the joint angles
 - Post-processing fix with IK

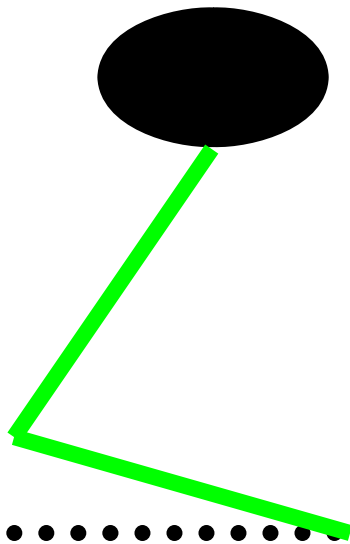


IK tricks:

- joint limits
- natural poses
- solution coherency

Blending Motions

1. Blend the joint angles
 - Post-processing fix with IK

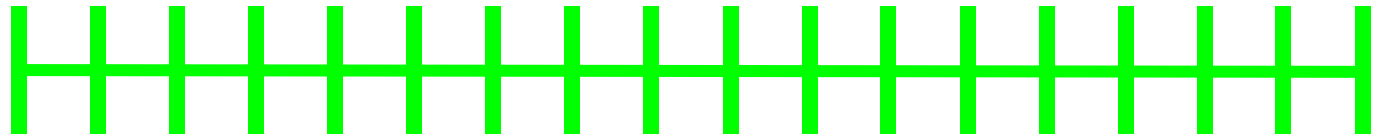


IK tricks:

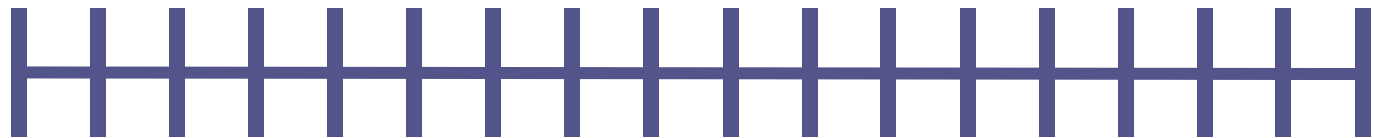
- joint limits
- natural poses
- solution coherency

Blending Motions

**Running
motion**

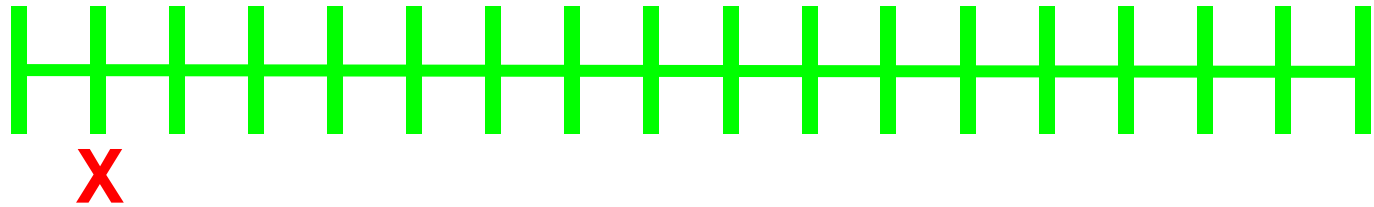


**Walking
motion**



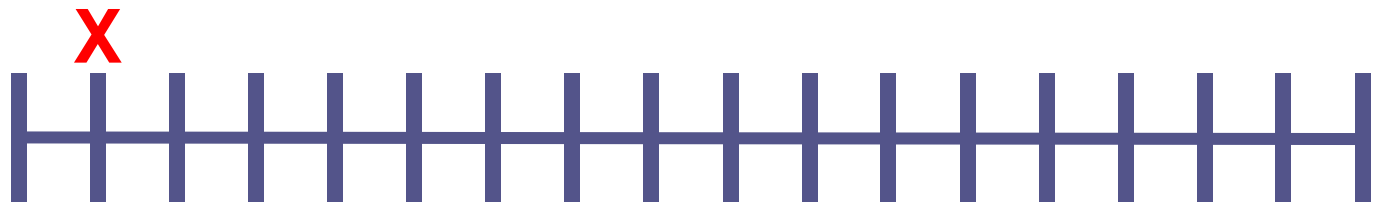
Blending Motions

**Running
motion**



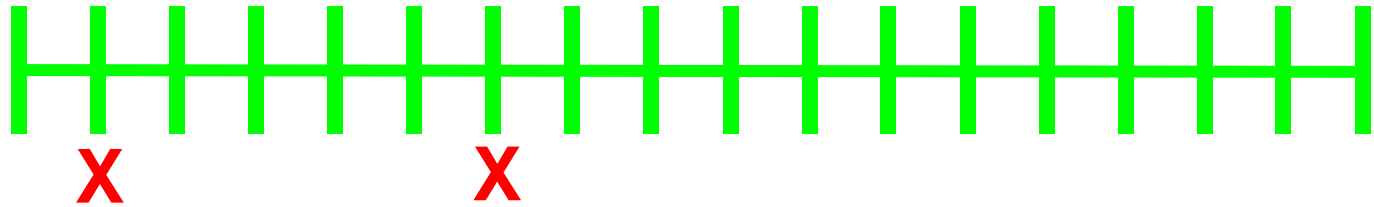
Footfalls

**Walking
motion**



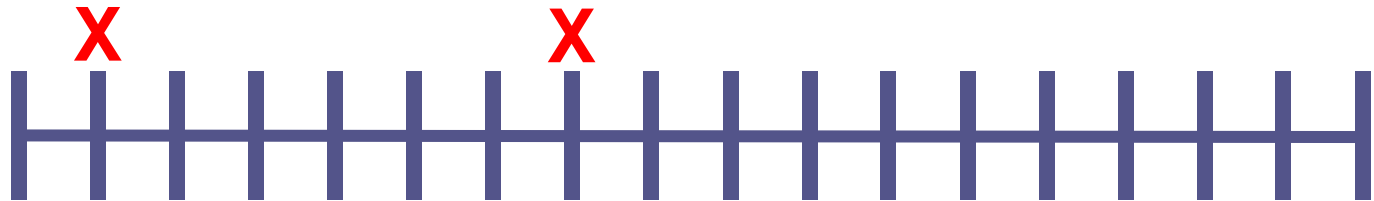
Blending Motions

**Running
motion**



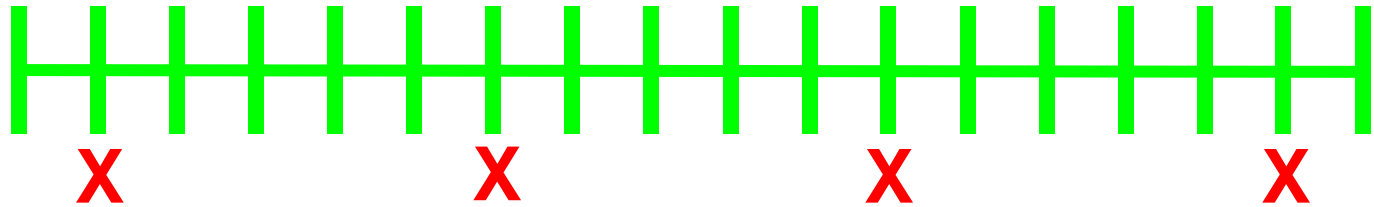
Footfalls

**Walking
motion**



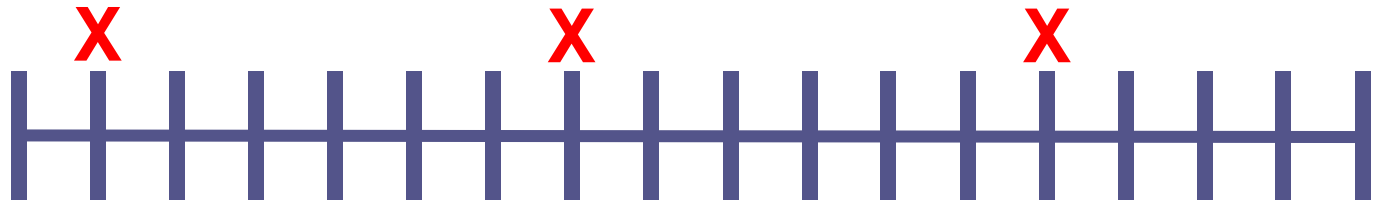
Blending Motions

Running motion



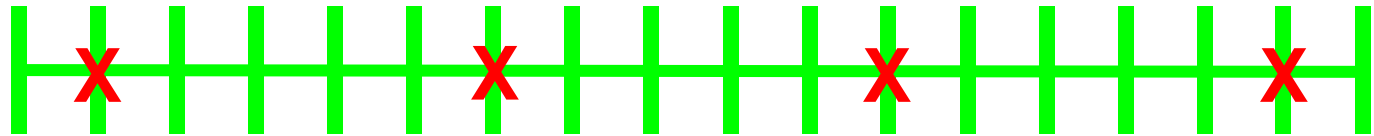
Footfalls

Walking motion

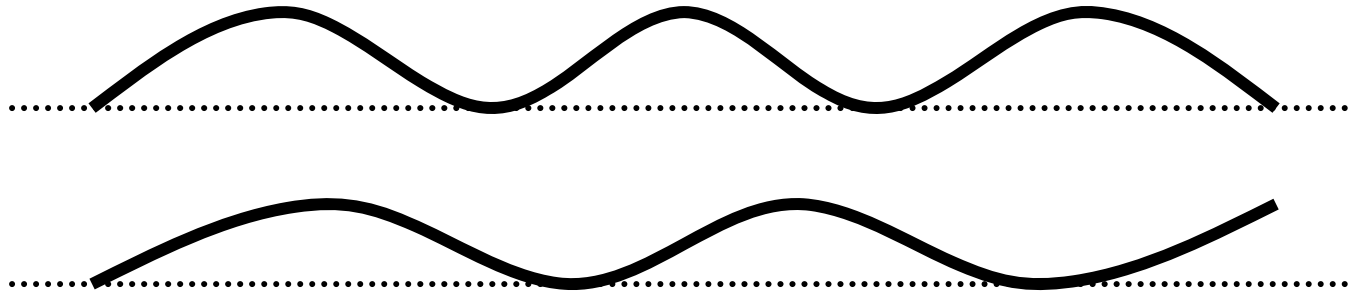


Blending Motions

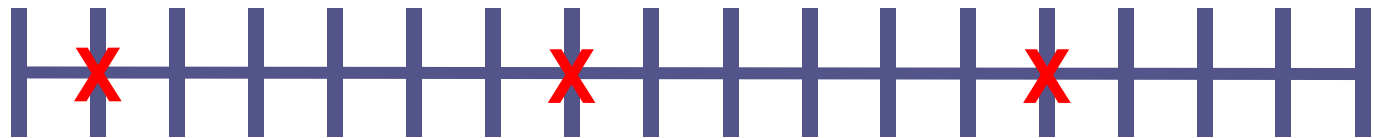
**Running
motion**



**Knee
angle**

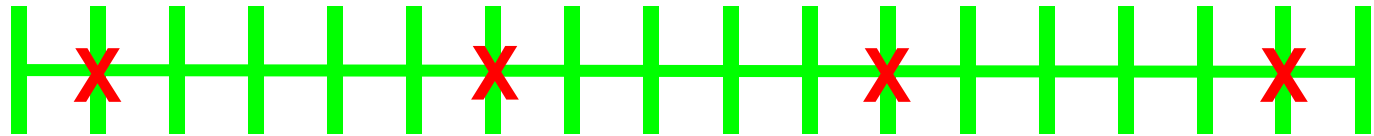


**Walking
motion**



Blending Motions

**Running
motion**



**Knee
angle**

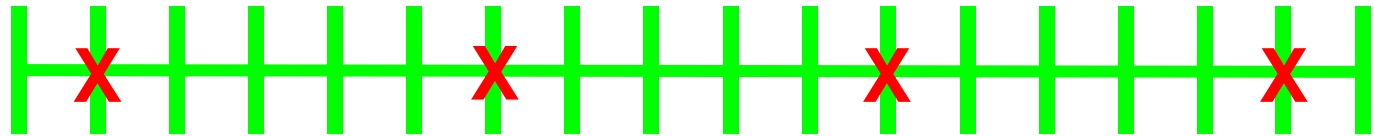


**Walking
motion**



Blending Motions

Running motion



Average knee angle



Walking motion



Blending Motions

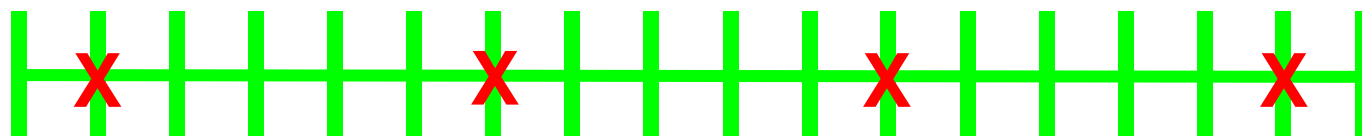
1. Blend the joint angles
 - Problem: even similar motions get out of phase
 - *Time alignment algorithm*

Time Alignment Algorithm

- Align logically similar parts of motion
 - e.g., footfalls with footfalls
 - Use *motion similarity metric*
- Define mapping between frames of motions
 - Shorter motion duplicates frames to match longer motion

Aligning Motions

**Running
motion**



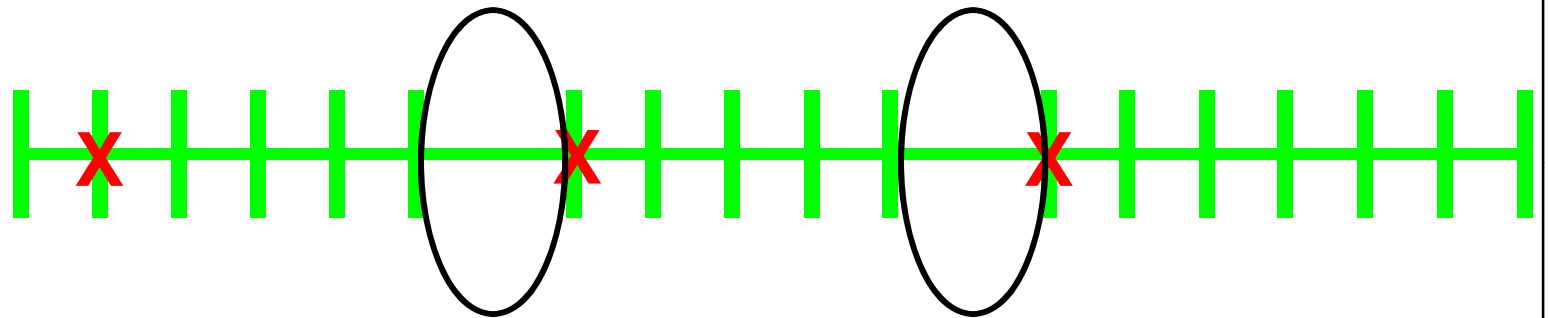
**Walking
motion**



Aligning Motions

- Duplicate frames to fill gaps

Running motion



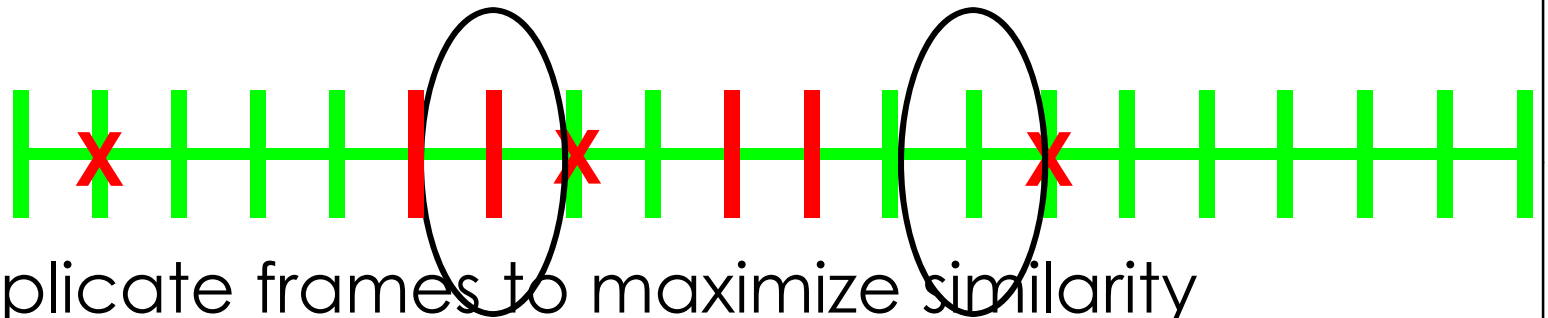
Walking motion



Aligning Motions

- Duplicate frames to fill gaps

Running motion



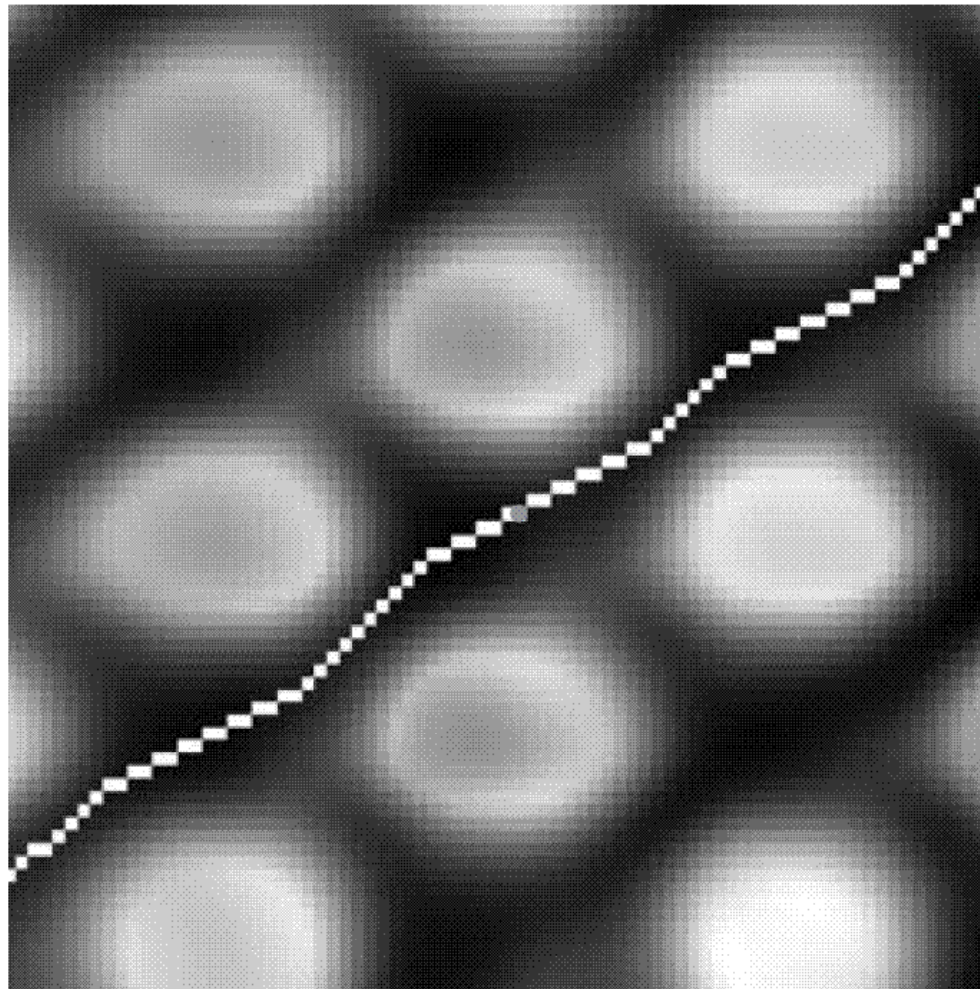
- Duplicate frames to maximize similarity

Walking motion



Dynamic Timewarping

Frame of walking motion

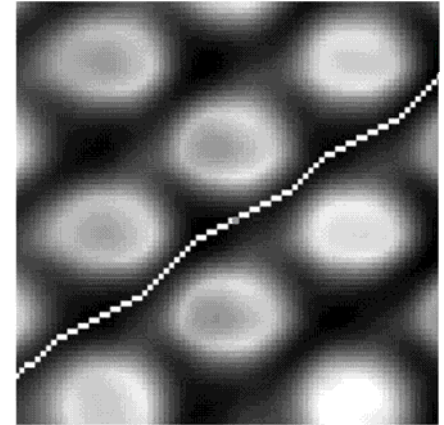


Frame of running motion

**Matrix of
frame-to-frame
similarity
(darker = more
similar)**

Dynamic Timewarping

- Choose path through matrix such that:
 - Path is continuous
 - Path is causal
 - Path has limited slope

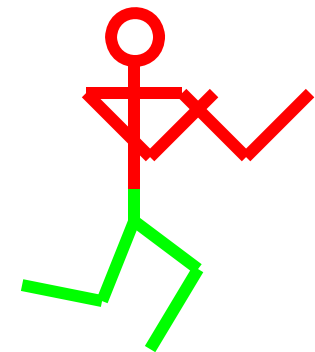


Transplanting Body Parts

- Have a running motion
- Have a catching motion
- Want a catching-while-running motion

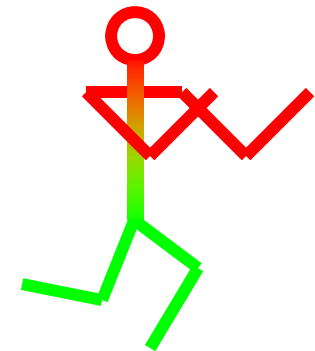
Transplanting Body Parts

- Have a running motion
- Have a catching motion
- Want a catching-while-running motion
- **Solution:** stick a catching torso on top of running legs



Transplanting Body Parts

- **Solution:** stick a catching torso on top of running legs
- Use *motion blending*
 - Joints in legs get 100% of angles from running motion
 - Joints in arms get 100% of angles from catching motion
 - Joints in between blend both motions, with weights depending on whether closer to legs or arms

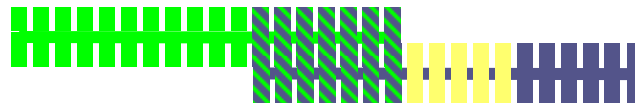


Motion Editing

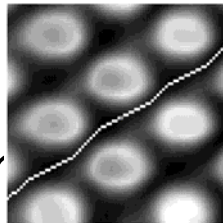
- Warping



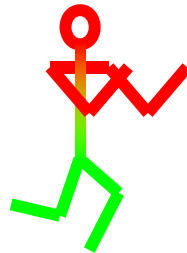
- Splicing



- Blending



- Transplanting



More Information

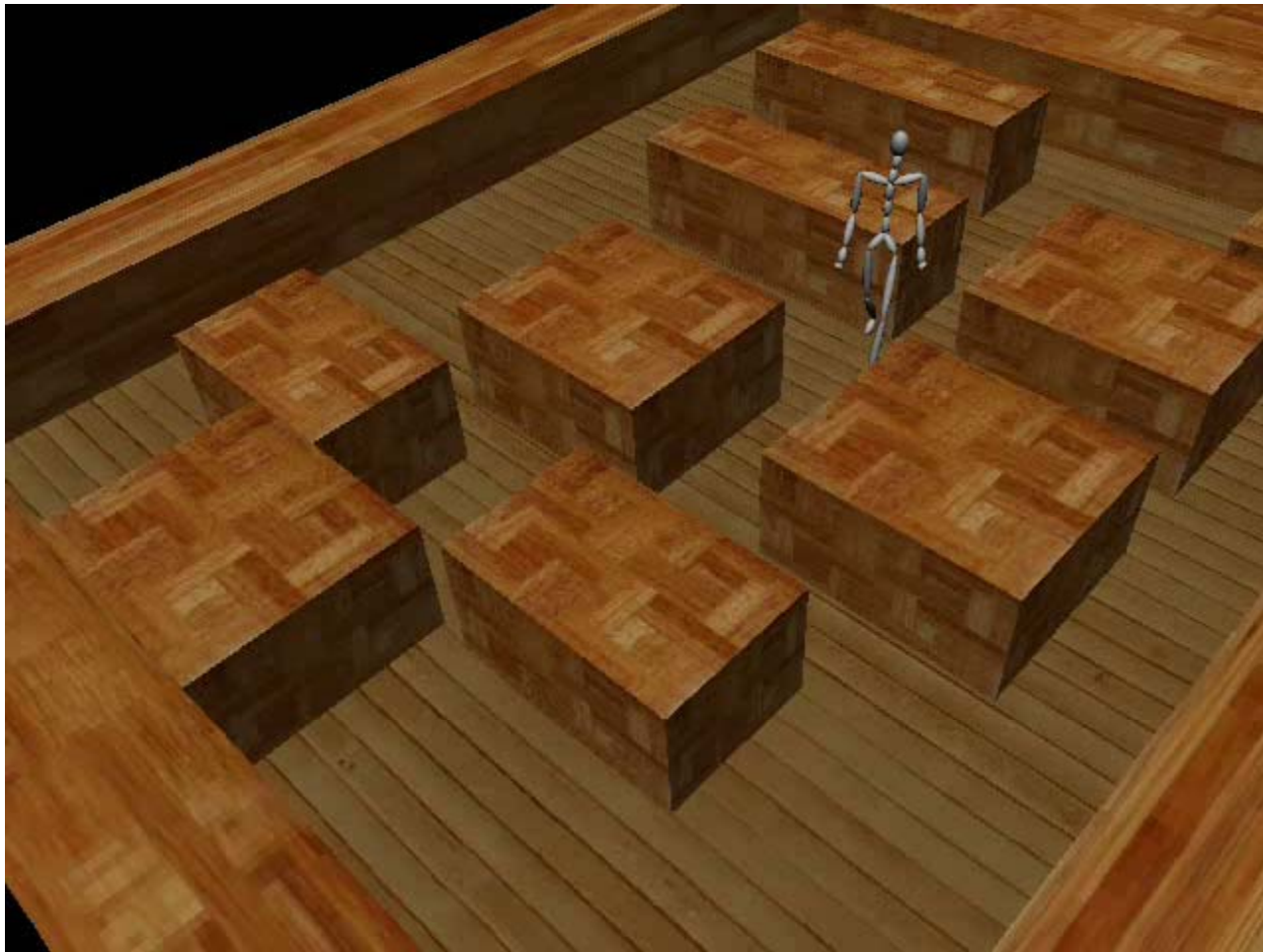
- Warping
 - “A hierarchical approach to interactive motion editing for human-like figures”, Lee and Shin, 1999
- Splicing
 - “Motion Graphs”, Kovar et al., 2002
- Blending
 - “Flexible Automatic Motion Blending with Registration Curves”, Kovar and Gleicher, 2003
- Transplanting
 - “Enriching a Motion Collection by Transplanting Limbs”, Ikemoto and Forsyth, 2004

Motion Graphs and Character Control

Motion Graphs

- Goal: synthesize animation for an unpredictable character
 - Screen saver
 - Game character
 - Training sim/freeform animation/prototyping/...

Motion Graphs



From Lee et al., Siggraph 2002

Example Motion Graph

- Source motions:

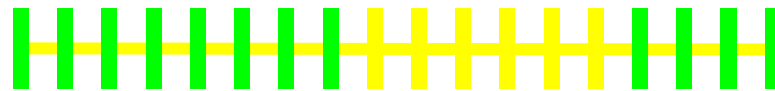
- Run:



- Duck:



- Punch:



.....→
Time

Step 1: Build Similarity Matrices

For each frame i in running motion

For each frame j in ducking motion

$$\begin{aligned} \text{cost}(i,j) = & \text{posecost}(i,j) \\ & + \text{velocitycost}(i,j) \\ & + \text{accelcost}(i,j) \end{aligned}$$

$$\text{similarity}(i,j) = 1/(1+\text{cost}(i,j))$$

Cost Functions

- $\text{posecost}(i,j) = \sum w(\text{joint}) * |\text{joint}(i) - \text{joint}(j)|$
 - $w(\text{hip}) = 1.0$
 - $w(\text{shoulder}) = 0.75$
 - $w(\text{knee}) = 0.25$
 - $w(\text{elbow}) = 0.25$
 - $w(\text{else}) = 0.0$
- i.e., use gross limb movement to distinguish between motion types

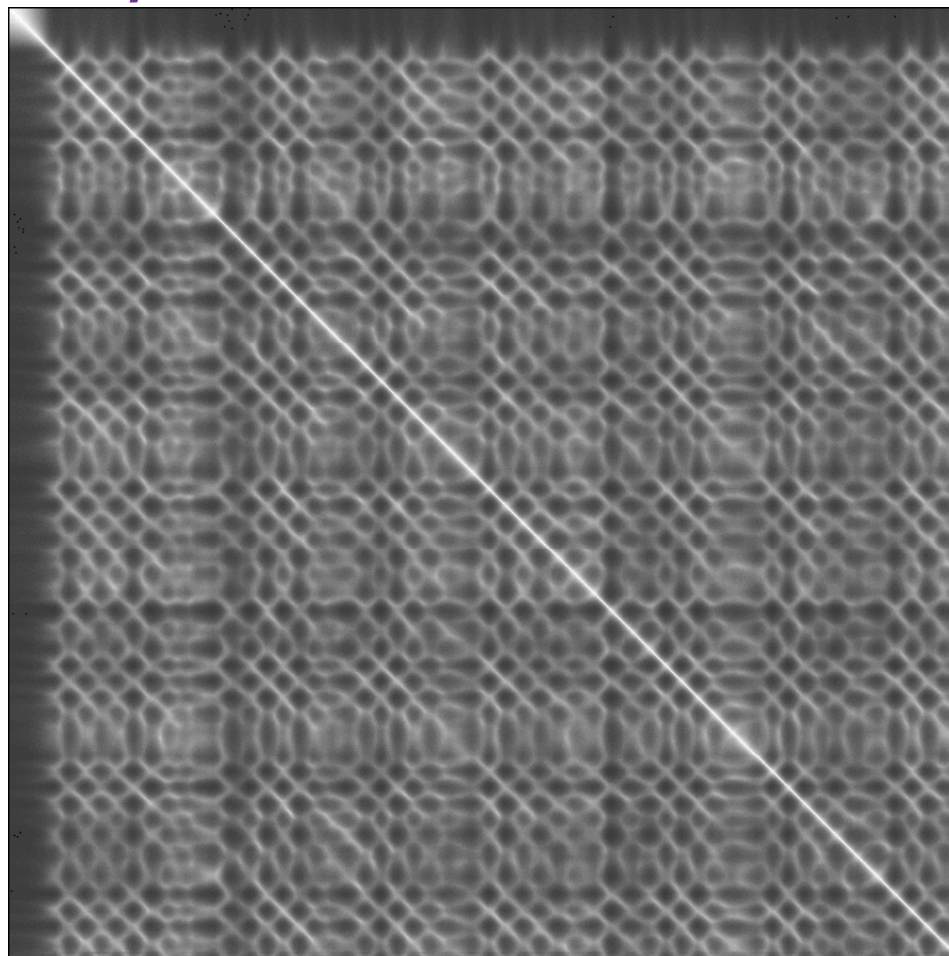
Cost Functions

- $\text{velocitycost}(i,j) = \sum w(\text{vel}) * w(\text{joint}) * | \text{jointvel}(i) - \text{jointvel}(j) |$
 - $\text{jointvel}(a) = \text{joint}(a) - \text{joint}(a-1)$
 - $w(\text{vel}) = 0.1$
- i.e., take into account the velocities of the key joints, but to a lesser extent than pose
 - Note: numerical differentiation is typically noisy

Cost Functions

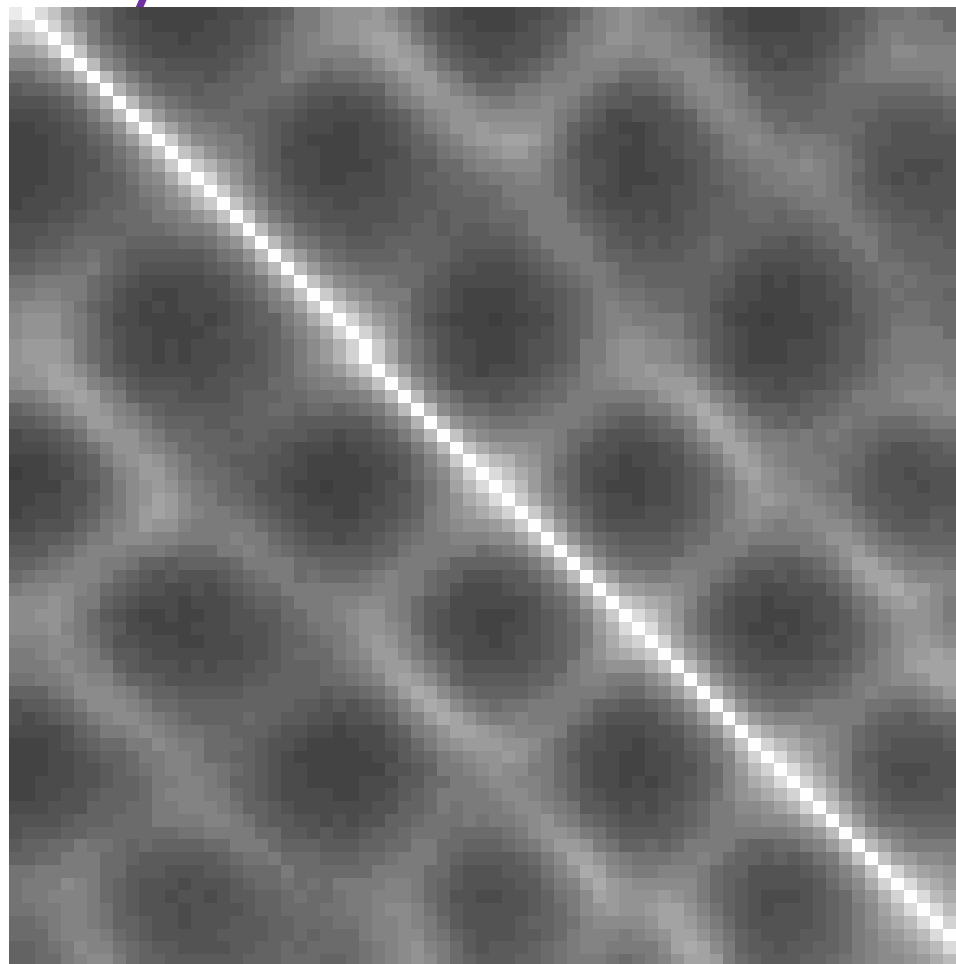
- $\text{accelcost}(i,j) = \sum w(\text{acc}) * w(\text{joint}) * | \text{jointacc}(i) - \text{jointacc}(j) |$
 - $\text{jointacc}(a) = \text{jointvel}(a) - \text{jointvel}(a-1)$
 $= \text{joint}(a) - 2 * \text{joint}(a-1) + \text{joint}(a-2)$
 - $w(\text{acc}) = 0.01$

Similarity Matrices



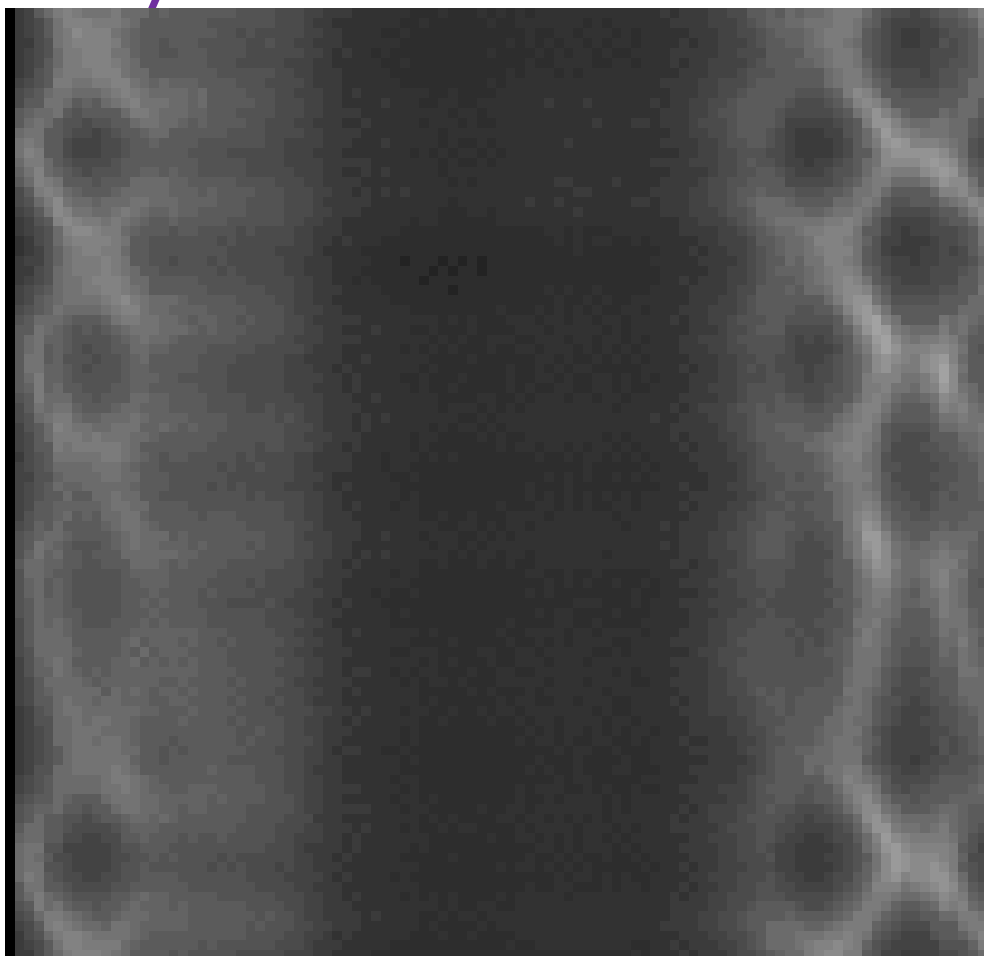
Running vs. Running

Similarity Matrices



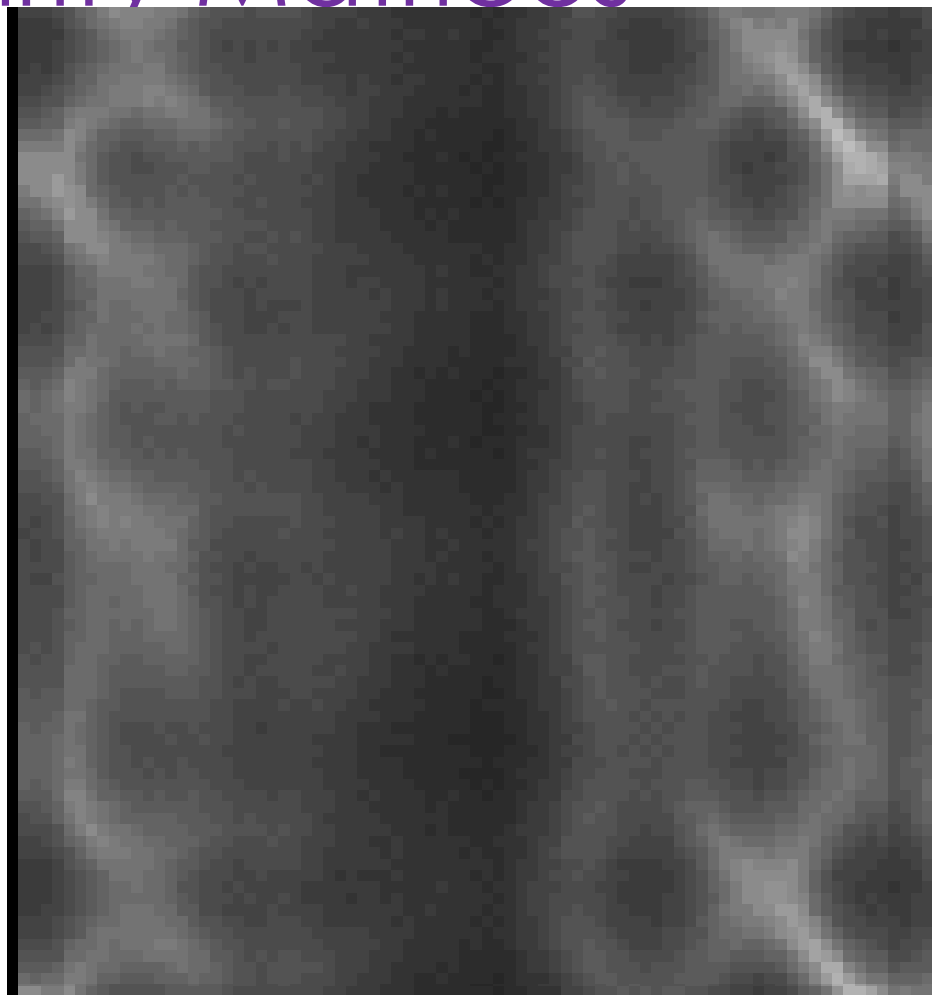
Running vs. Running

Similarity Matrices



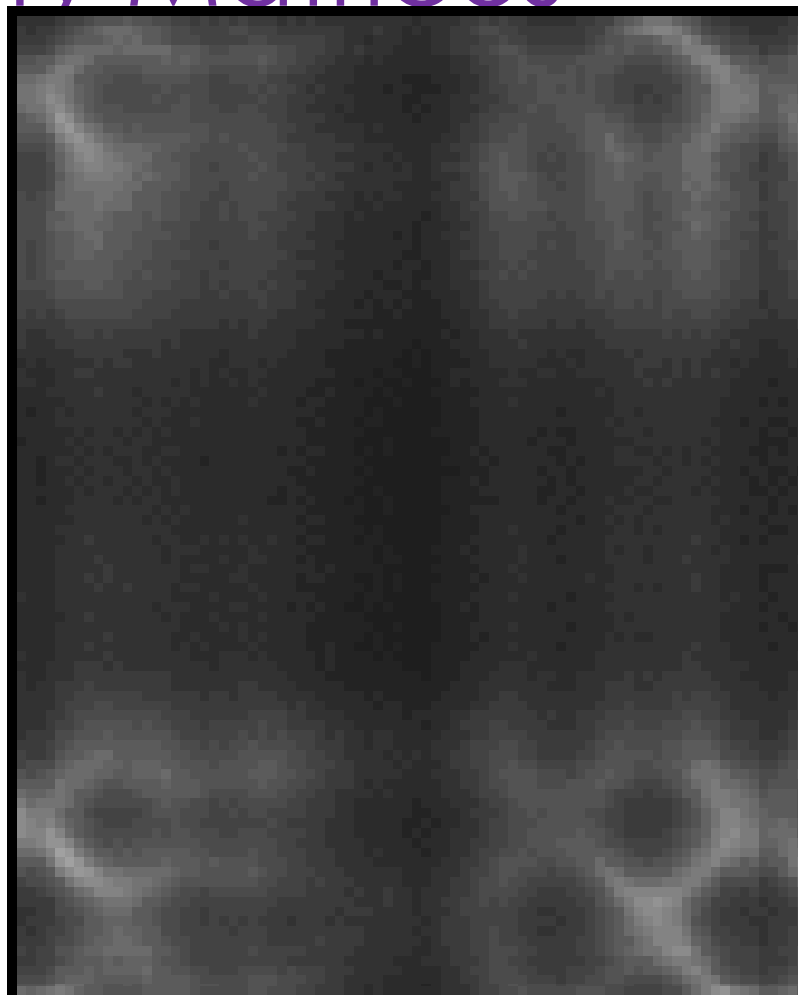
Running vs. Ducking

Similarity Matrices



Running vs. Punching

Similarity Matrices

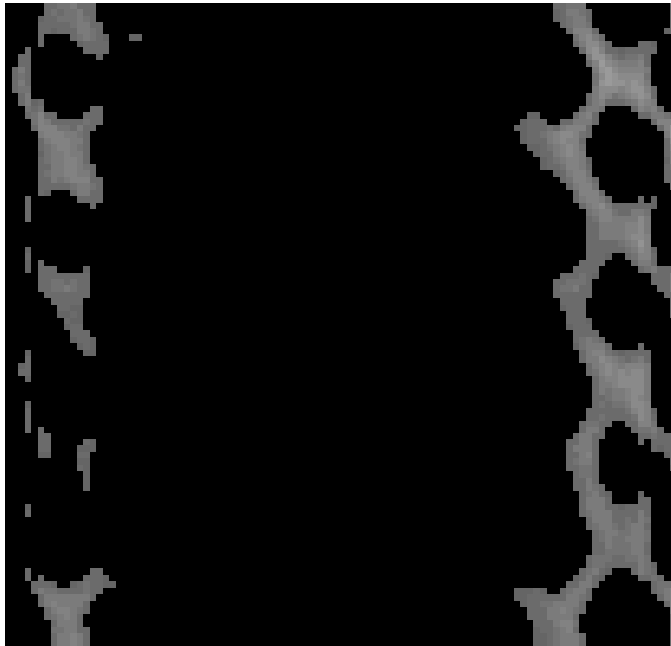


Ducking vs. Punching

Similarity Matrices

- Motion types split up despite simple metric
- Need a (small) representative set
 - Throw away anything under a threshold

Thresholded Matrices



Threshold 0.7



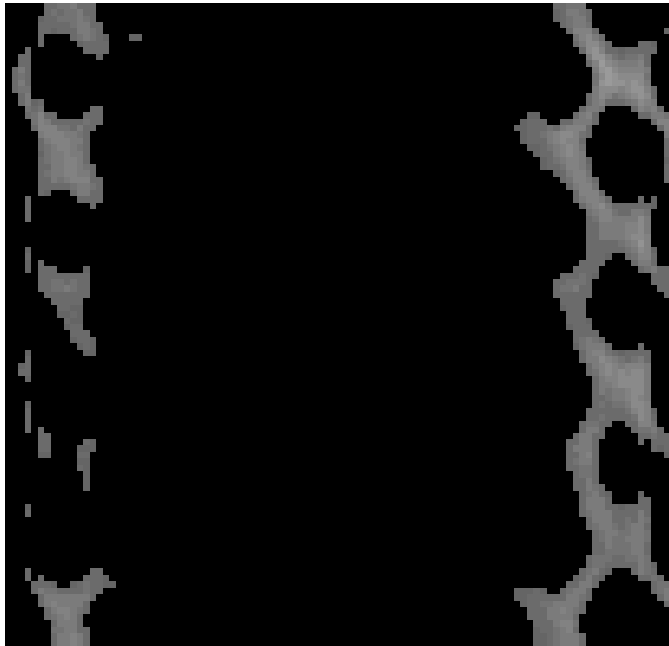
Threshold 0.9

Running vs. Ducking

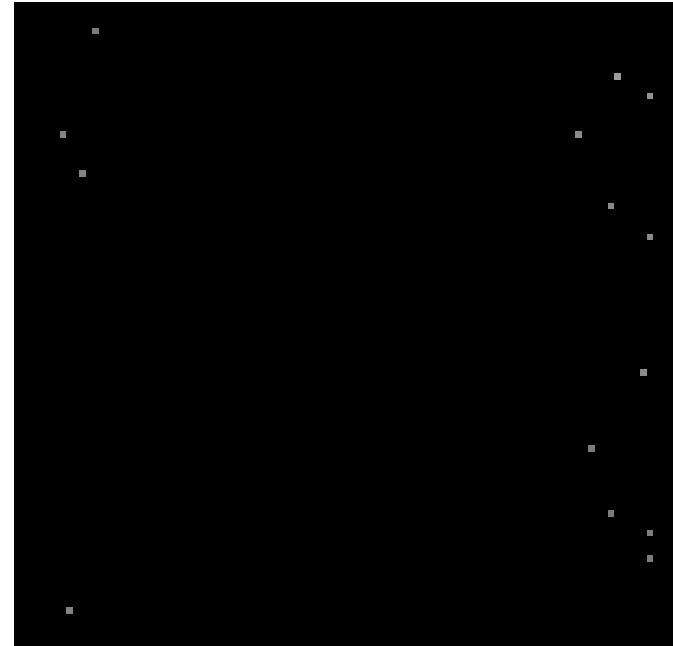
Similarity Matrices

- High-quality motions tend to cluster
 - Near-transitivity
- Want only a *representative* of each cluster
 - Remove values which are not local maxima

Thresholded Matrices



Threshold 0.7



Local Maxima

Running vs. Ducking

Graph Structure

- Similarity matrices give list of frame equivalencies
 - e.g., $\text{run}(5) = \text{duck}(7)$
- Need list of transitions can take
 - Remember flow of time
 - $\text{run}(5) \rightarrow \text{duck}(8)$
 - $\text{duck}(7) \rightarrow \text{run}(6)$
- Also original transitions
 - $\text{run}(5) \rightarrow \text{run}(6)$

De-seam Transitions

- New transitions have small gaps
- Quaternion smoothing
 - Convert angles to quaternions
 - Determine offset between A(last) and B(first)
 - Take away offset over N frames
 - SLERP(B(t), B(t)+offset, 1-i/N)

Basic Motion Graph

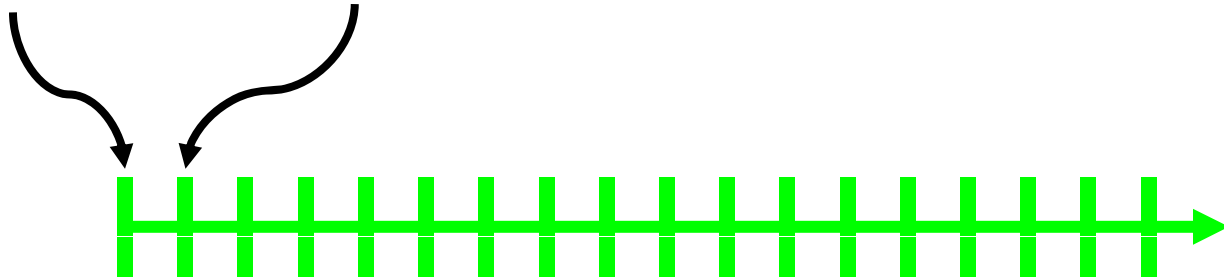
1. Acquire motion
2. Create similarity matrices
3. Cull matrices to local maxima, threshold
4. Build graph
5. De-seam transitions at runtime

Motion Graph Benefits

- Re-use of high-quality input
 - Minimal editing, so maintain characteristics
- Automatic
 - Large amounts of data
 - Novice users
 - Natural transitions between behaviours
 - As-good-as-possible added transitions
- Fast and easy to use at runtime

Motion Graphs

Frame 1 Frame 2 ...



Motion 1

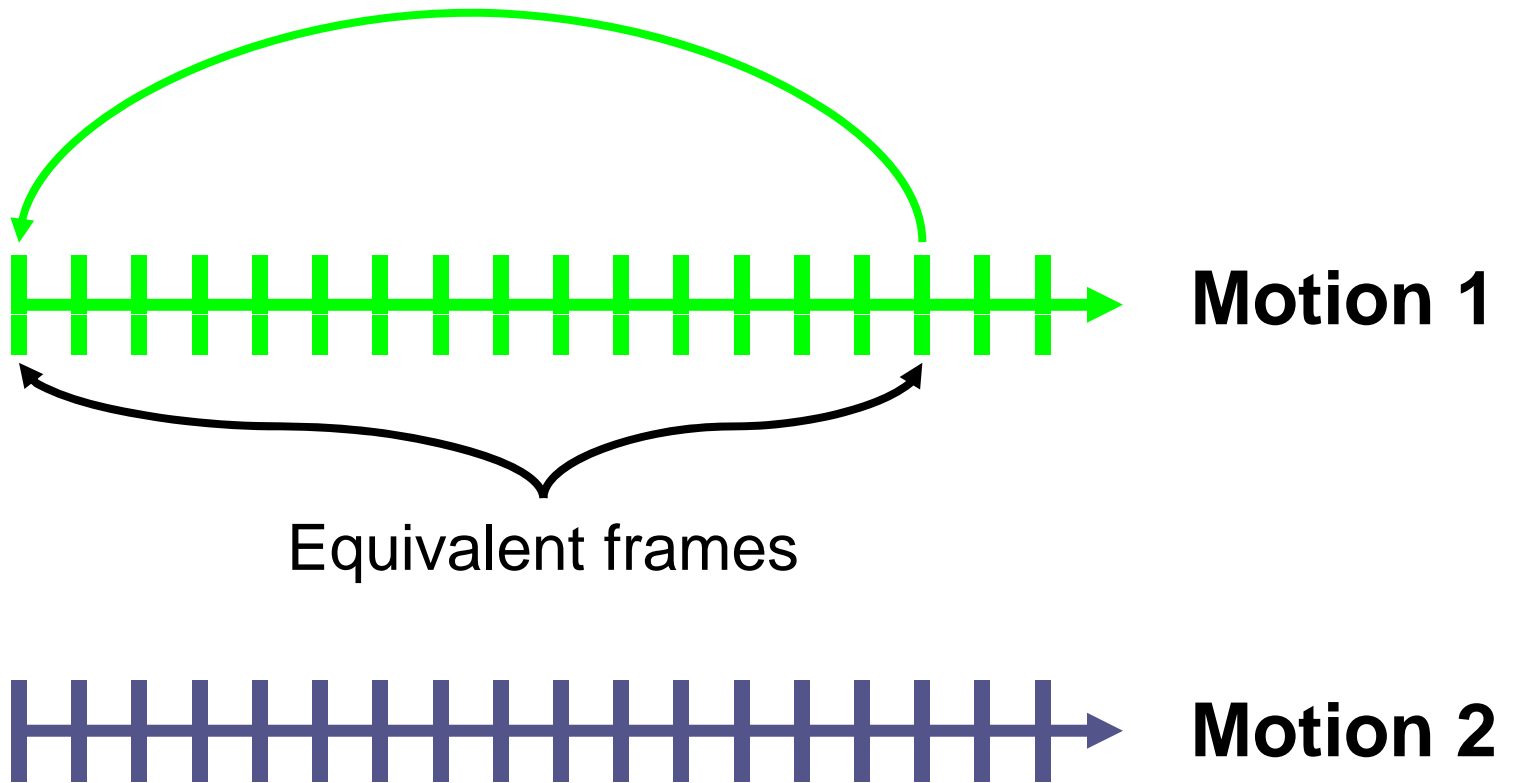


Motion 2

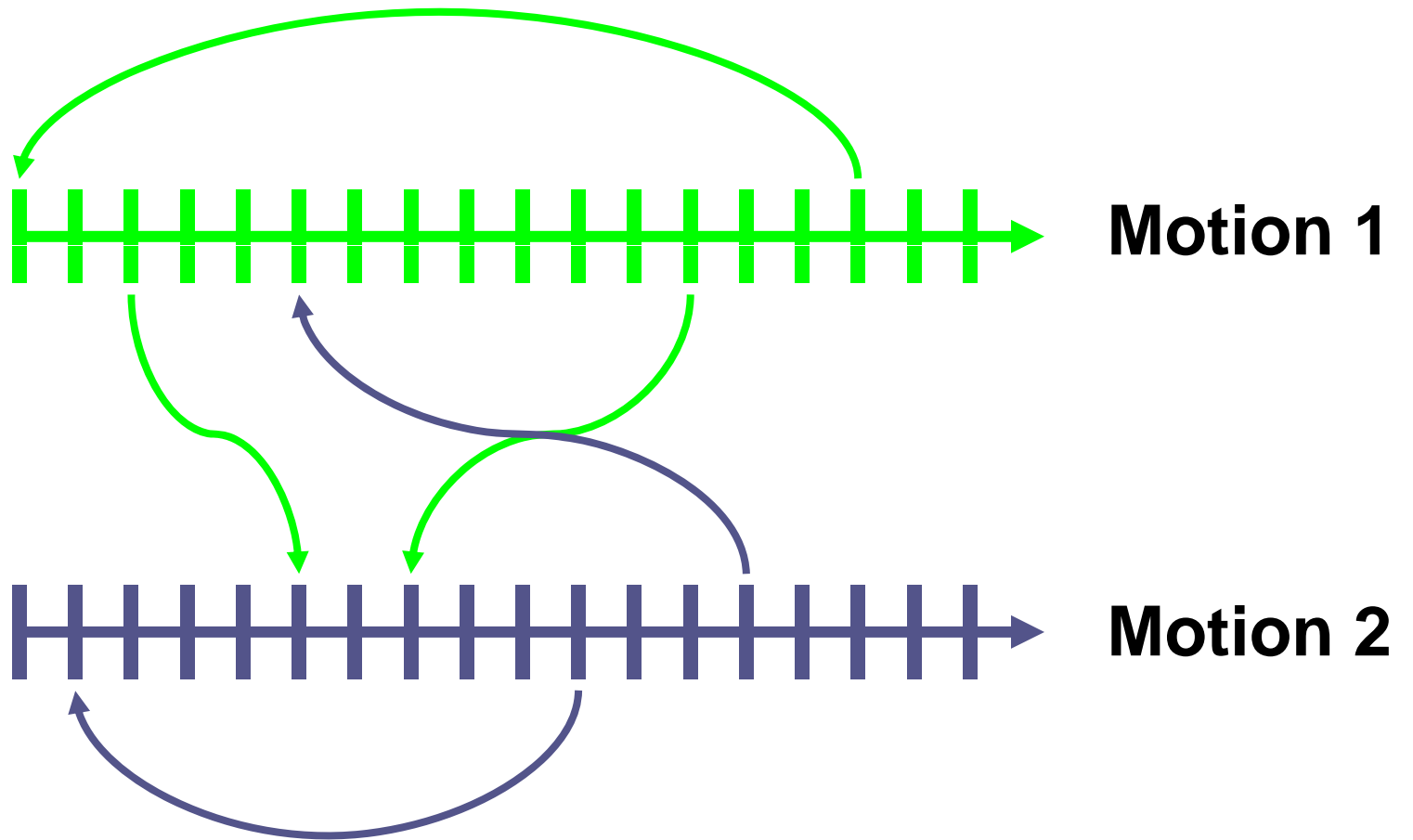
Time



Motion Graphs



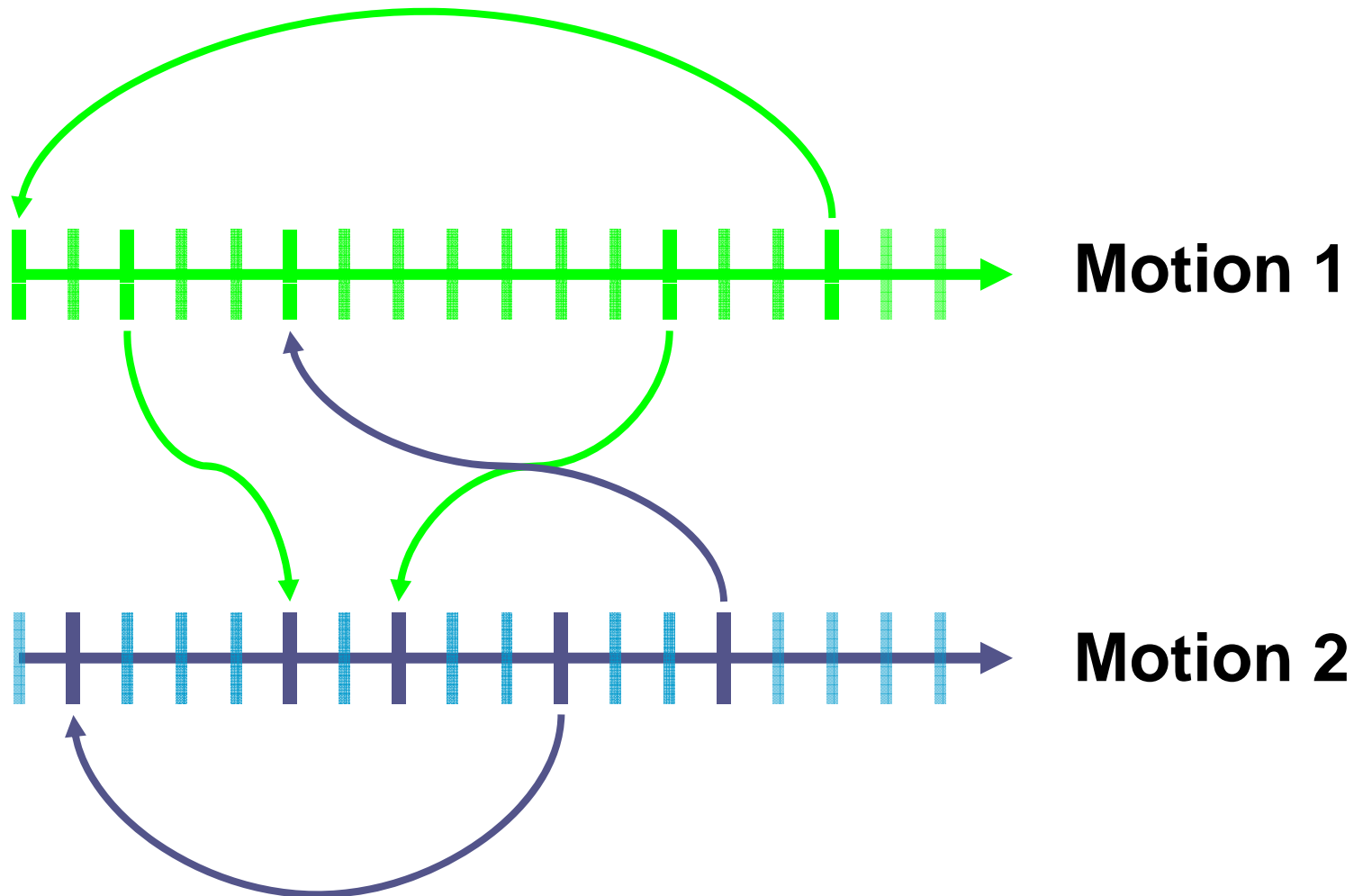
Motion Graphs



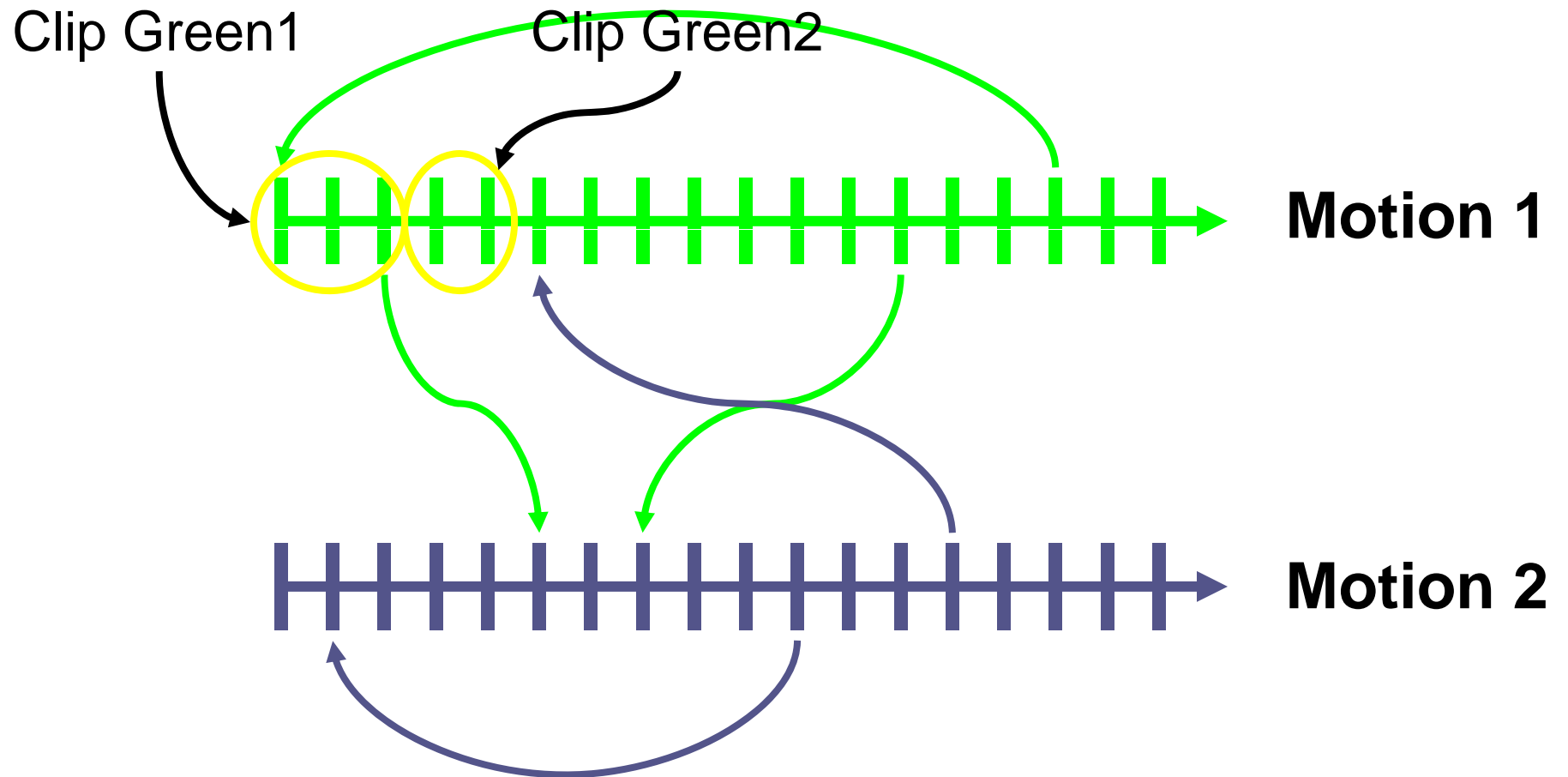
Motion 1

Motion 2

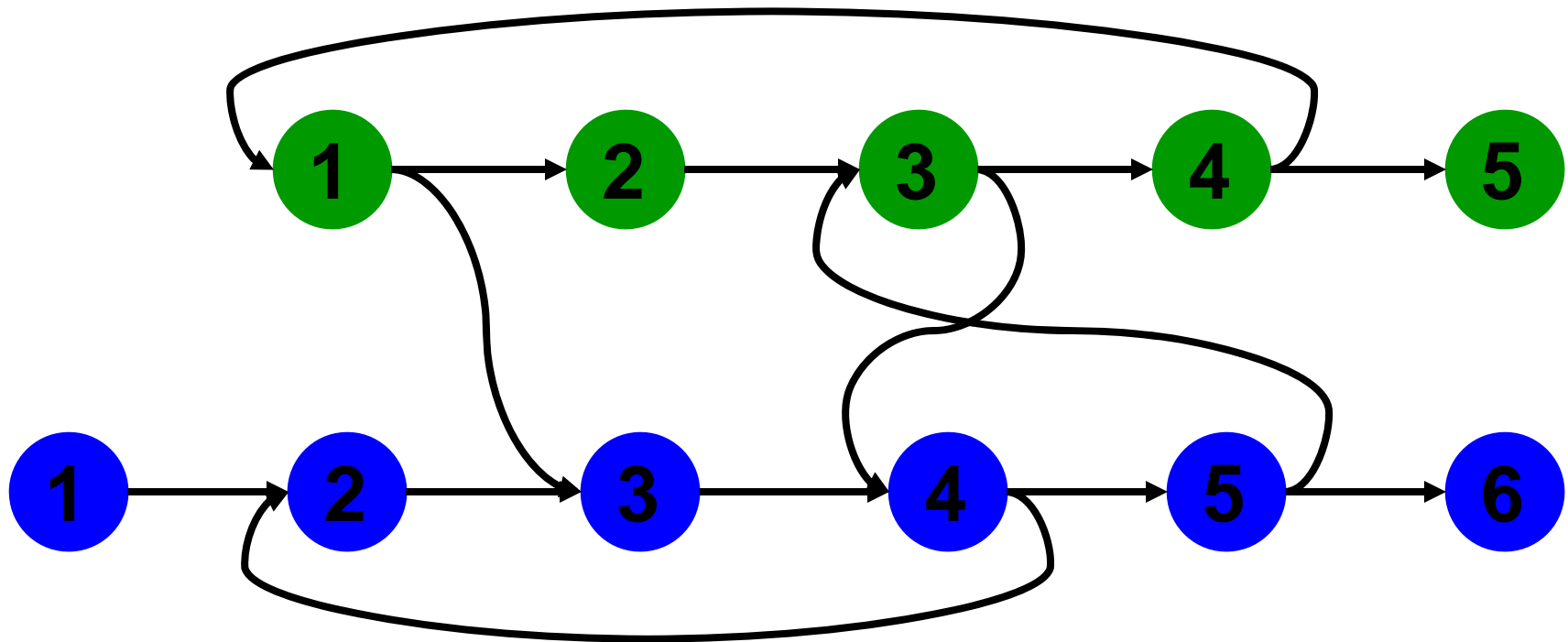
Motion Graphs



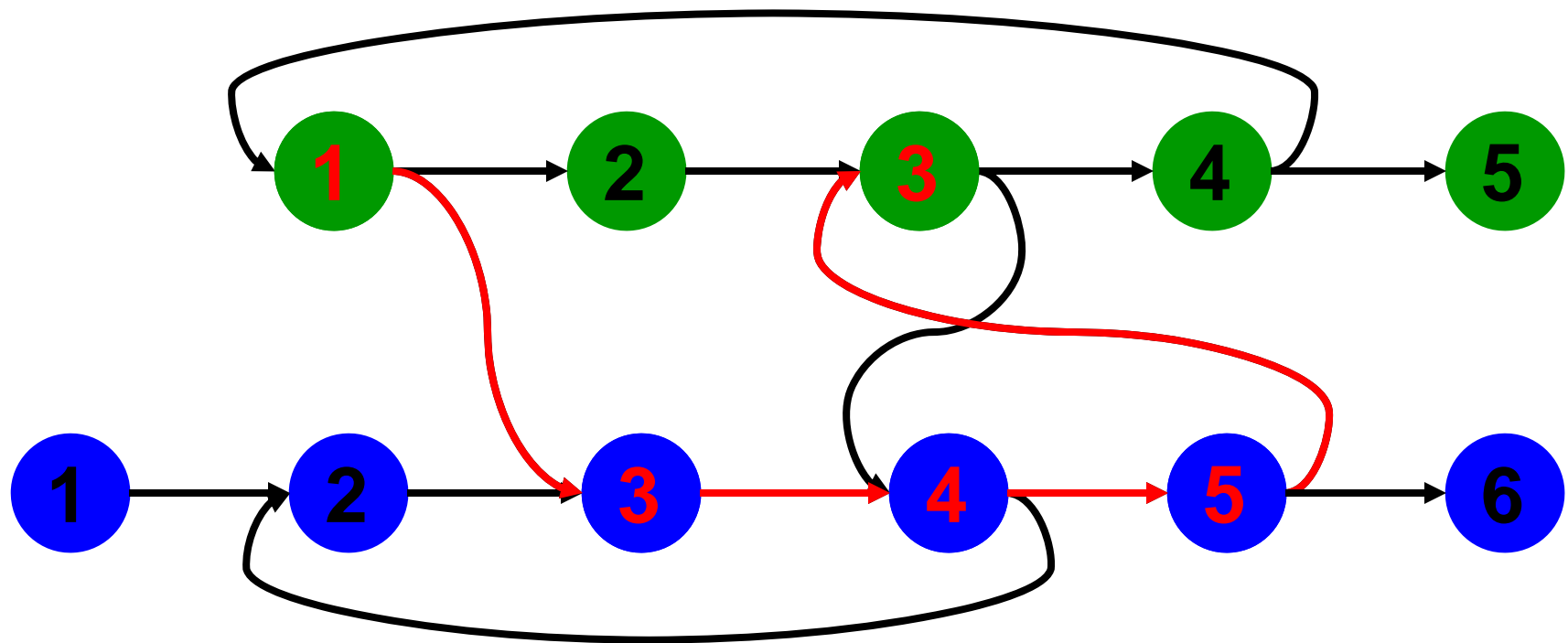
Motion Graphs



Motion Graphs



Motion Graphs



Generate animation by moving through graph

Sample Motion Graph

- Source motions:

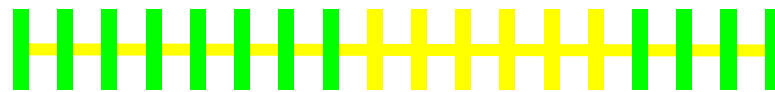
- Run:



- Duck:



- Punch:



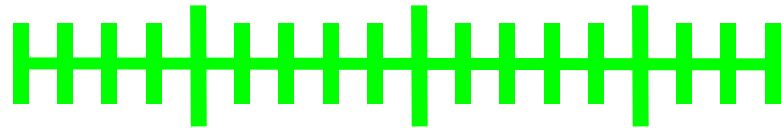
.....→
Time

Sample Motion Graph

- Equivalency list
 - $\text{run}(3) = \text{run}(15)$
 - $\text{run}(3) = \text{duck}(2)$
 - $\text{run}(8) = \text{duck}(15)$
 - $\text{run}(6) = \text{punch}(3)$
 - $\text{run}(11) = \text{punch}(17)$

Example Motion Graph

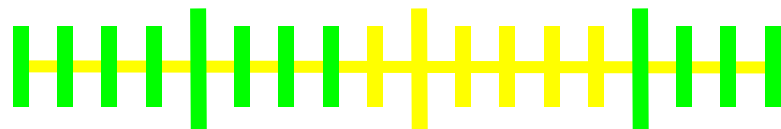
Run



Duck



Punch

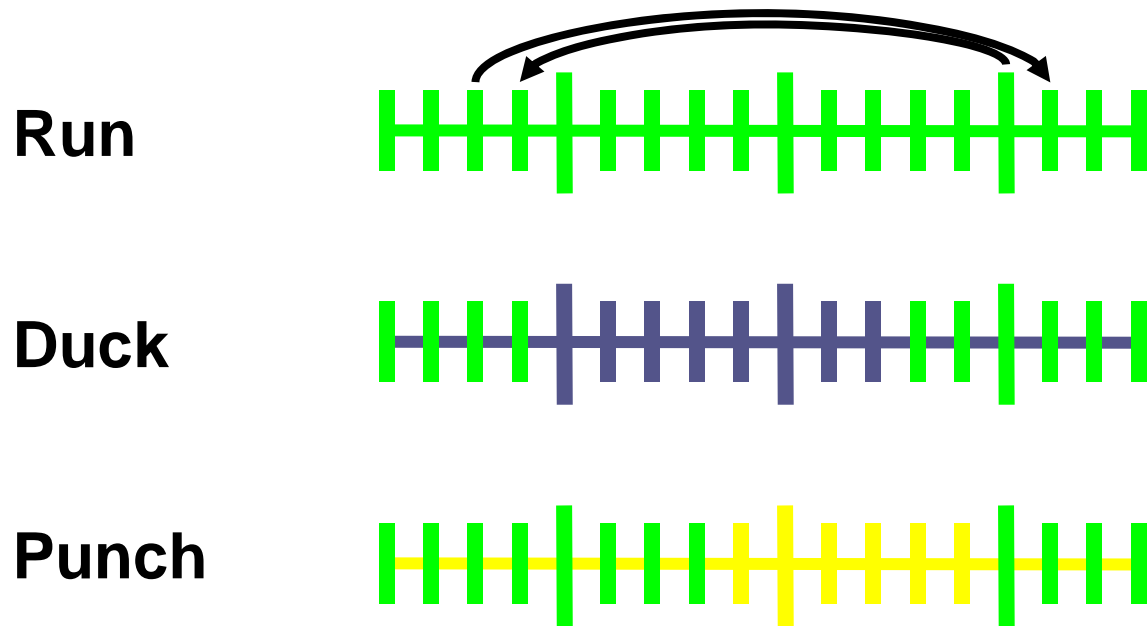


■ Equivalency list:

- run(3) = run(15)
- run(3) = duck(2)
- run(8) = duck(15)

- run(6) = punch(3)
- run(11) = punch(17)

Example Motion Graph

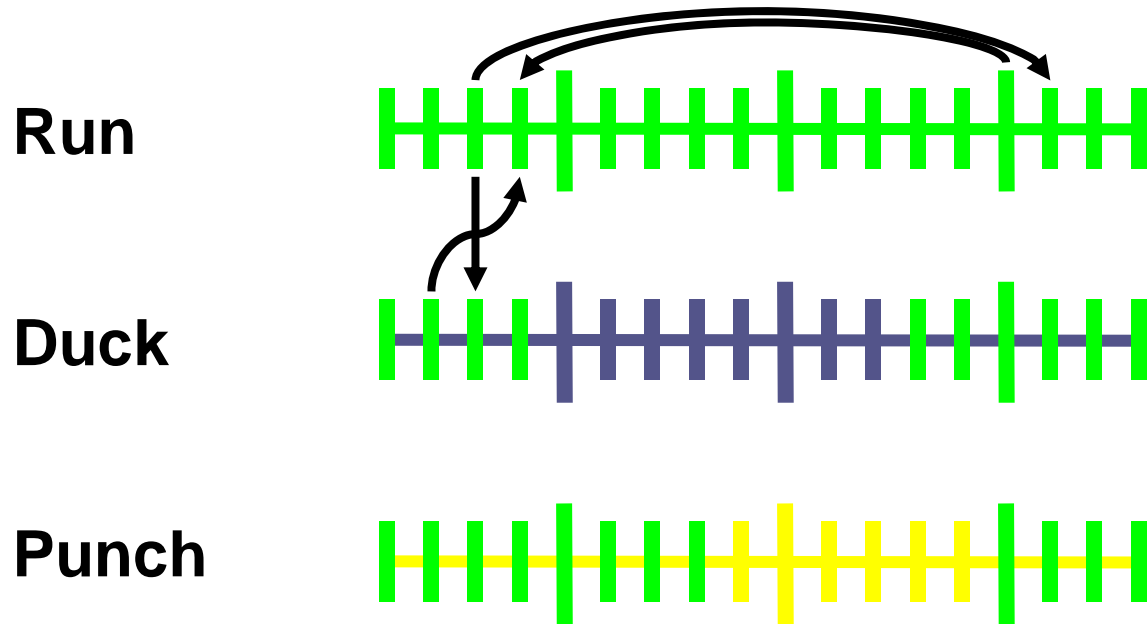


■ Equivalency list:

- $\text{run}(3) = \text{run}(15)$
- $\text{run}(3) = \text{duck}(2)$
- $\text{run}(8) = \text{duck}(15)$

- $\text{run}(6) = \text{punch}(3)$
- $\text{run}(11) = \text{punch}(17)$

Example Motion Graph

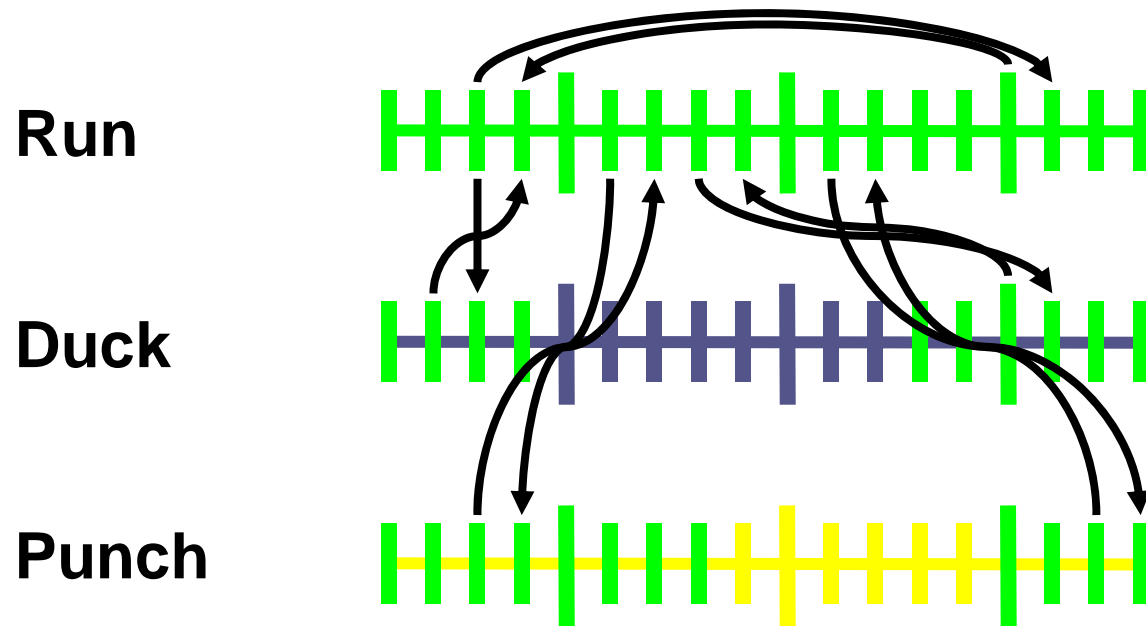


■ Equivalency list:

- run(3) = run(15)
- run(3) = duck(2)
- run(8) = duck(15)

- run(6) = punch(3)
- run(11) = punch(17)

Example Motion Graph



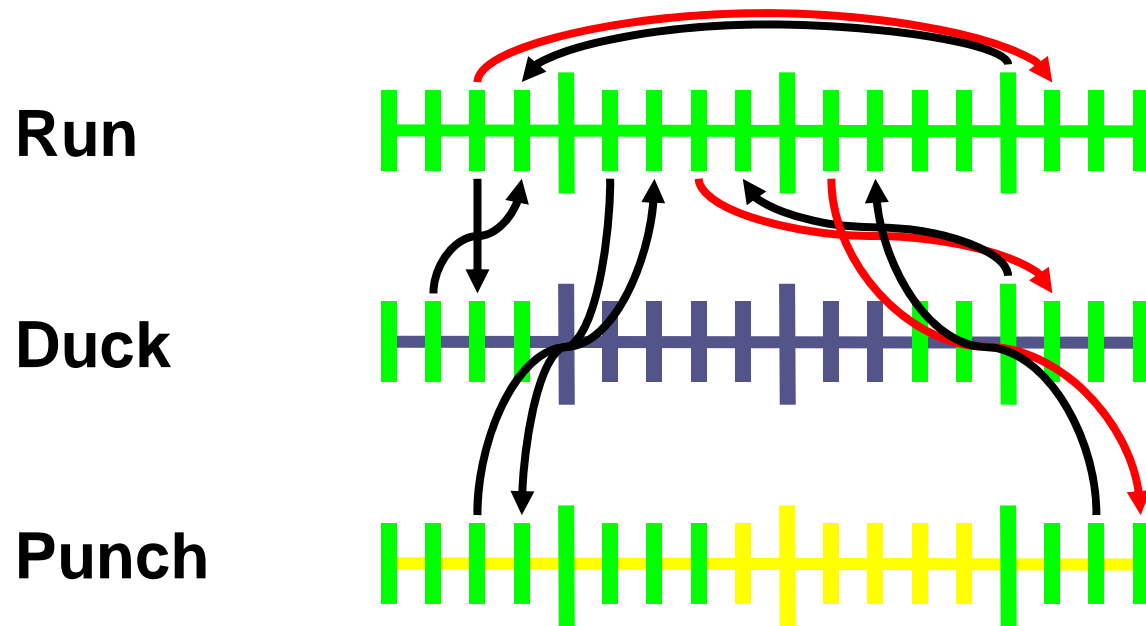
■ Equivalency list:

- run(3) = run(15)
- run(3) = duck(2)
- run(8) = duck(15)
- run(6) = punch(3)
- run(11) = punch(17)

Sample Motion Graph

- Recall: goal is synthesizing *ongoing* motion
 - Reaching the end of a clip means the animation stops
 - Cull out dead ends (SCC)

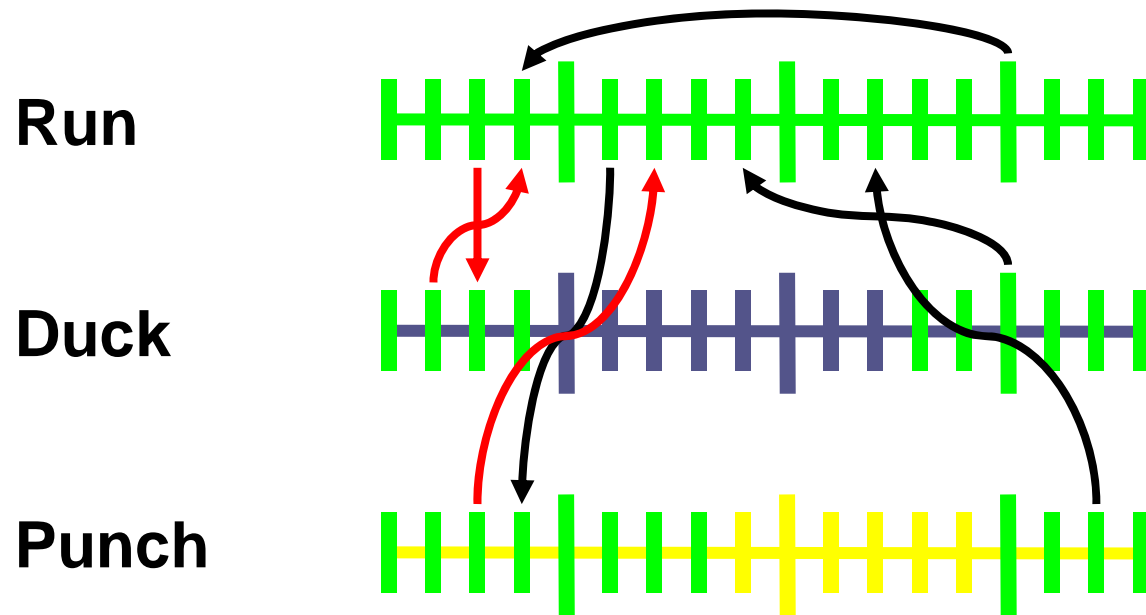
Example Motion Graph



■ Equivalency list:

- run(3) = run(15)
- run(3) = duck(2)
- run(8) = duck(15)
- run(6) = punch(3)
- run(11) = punch(17)

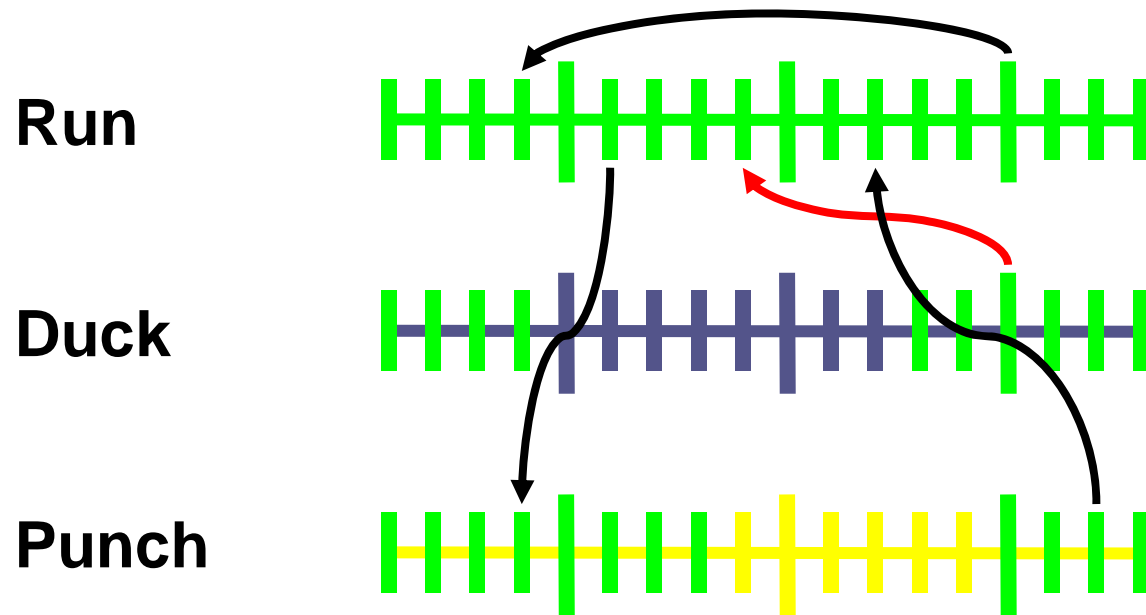
Example Motion Graph



■ Equivalency list:

- run(3) = run(15)
- run(3) = duck(2)
- run(8) = duck(15)
- run(6) = punch(3)
- run(11) = punch(17)

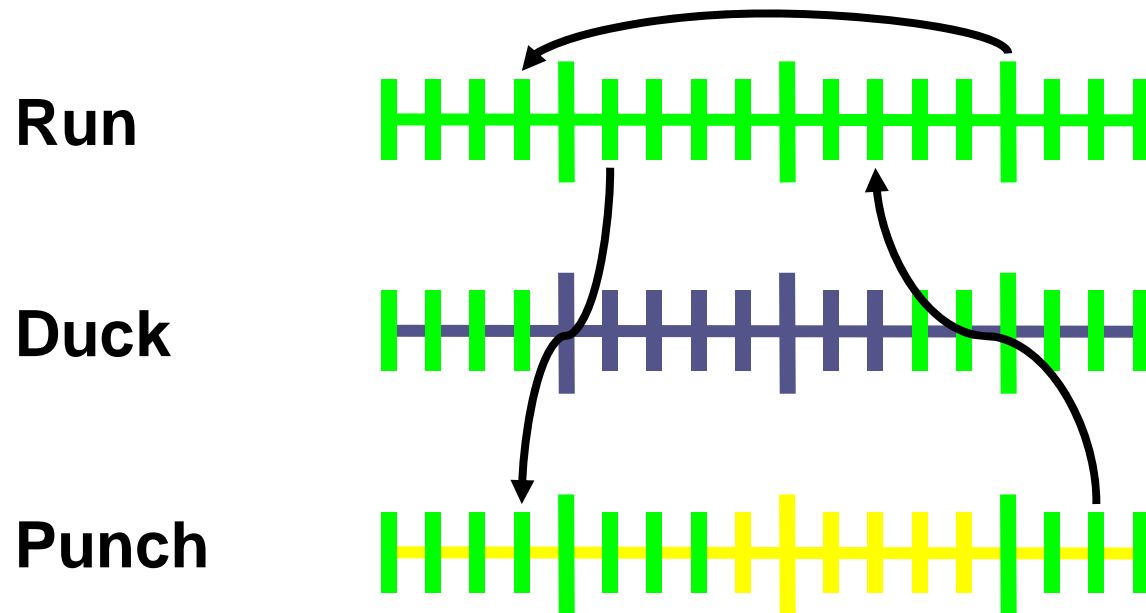
Example Motion Graph



■ Equivalency list:

- run(3) = run(15)
- run(3) = duck(2)
- run(8) = duck(15)
- run(6) = punch(3)
- run(11) = punch(17)

Example Motion Graph

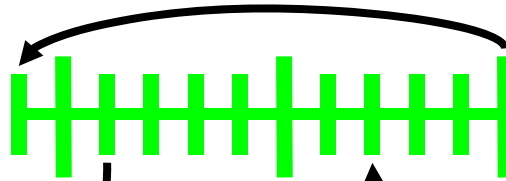


■ Equivalency list:

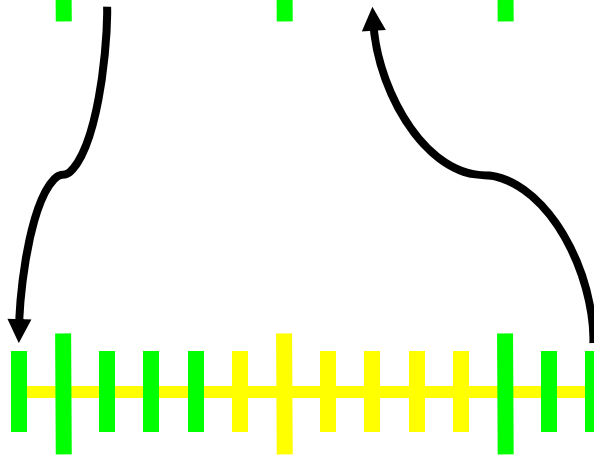
- run(3) = run(15)
- run(3) = duck(2)
- run(8) = duck(15)
- run(6) = punch(3)
- run(11) = punch(17)

Example Motion Graph

Run

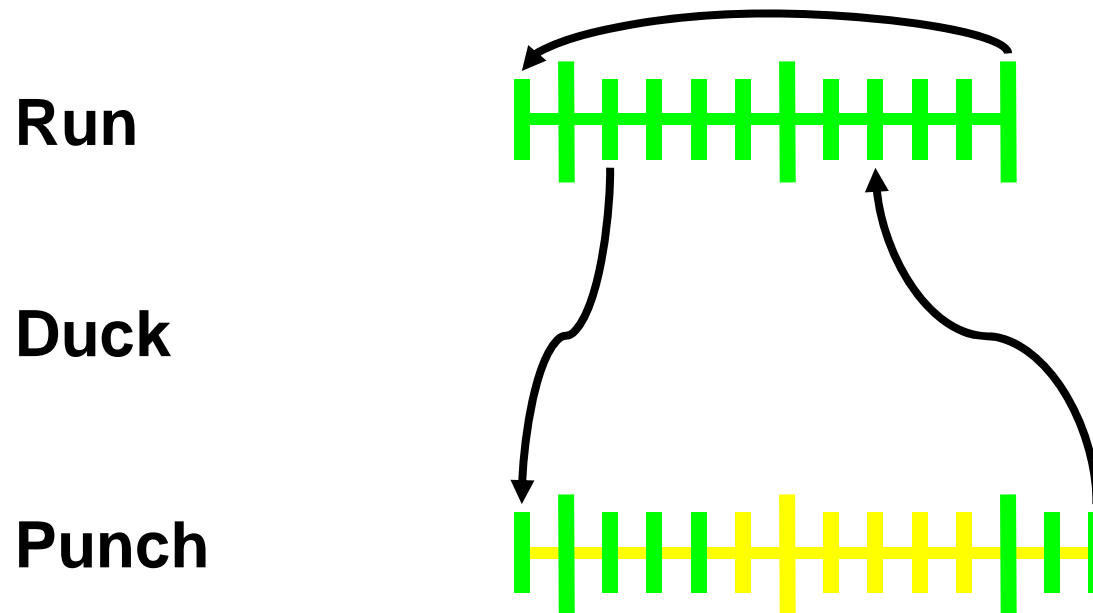


Duck



Punch

Example Motion Graph



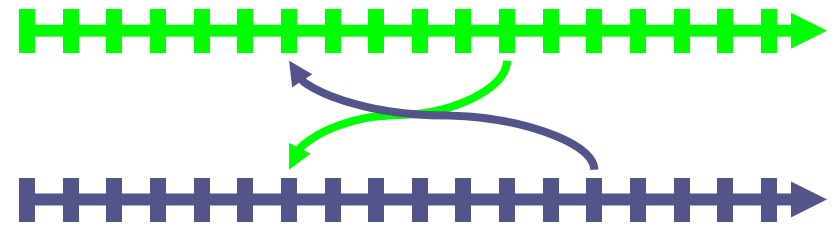
■ Notice:

- Clips end with an outgoing edge
- Clips start with an incoming edge
- The final graph is hard to predict

Motion Graphs

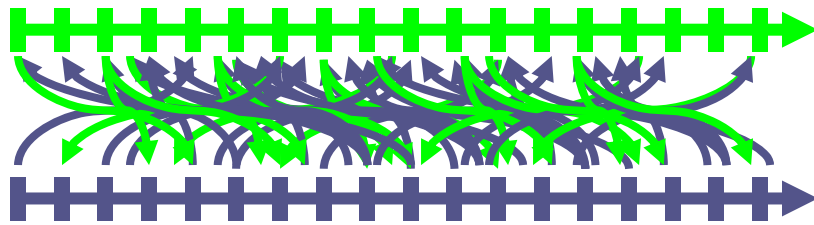
- Problem: hard to get a specific result

Standard Motion Graph Creation

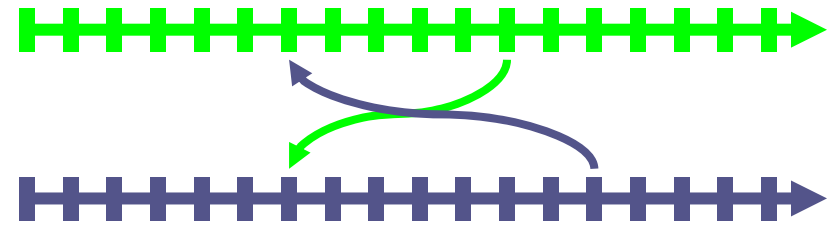


**High threshold:
few transitions**

Standard Motion Graph Creation

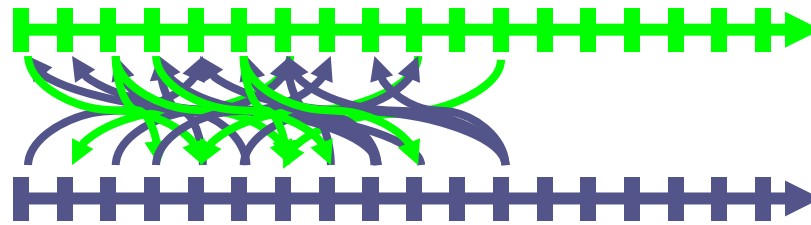


**Low threshold:
too many transitions**



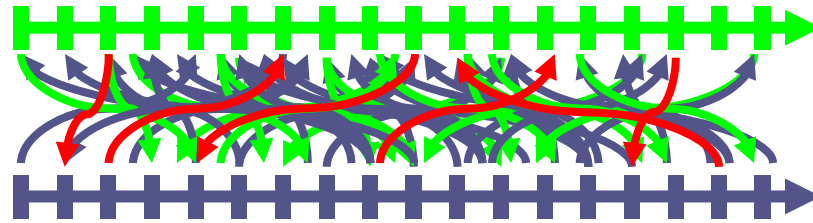
**High threshold:
too few transitions**

Standard Motion Graph Creation



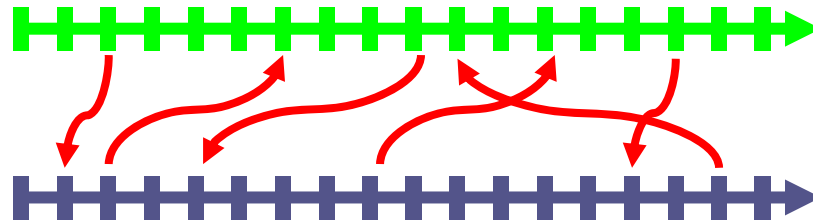
Medium threshold:
too few *and* too many transitions

Motion Graph Optimization



Optimize via evaluation-guided selection

Motion Graph Optimization



Optimize via evaluation-guided selection

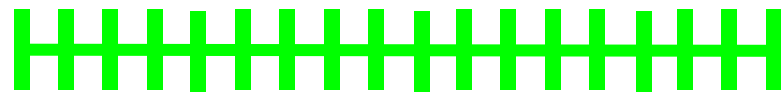
Motion Graphs

- Problem: hard to get a specific result
 - Optimizable
 - Or just tweak it. A lot.
- Other problems?

Jumping Puzzle

- Suppose you have a motion graph...

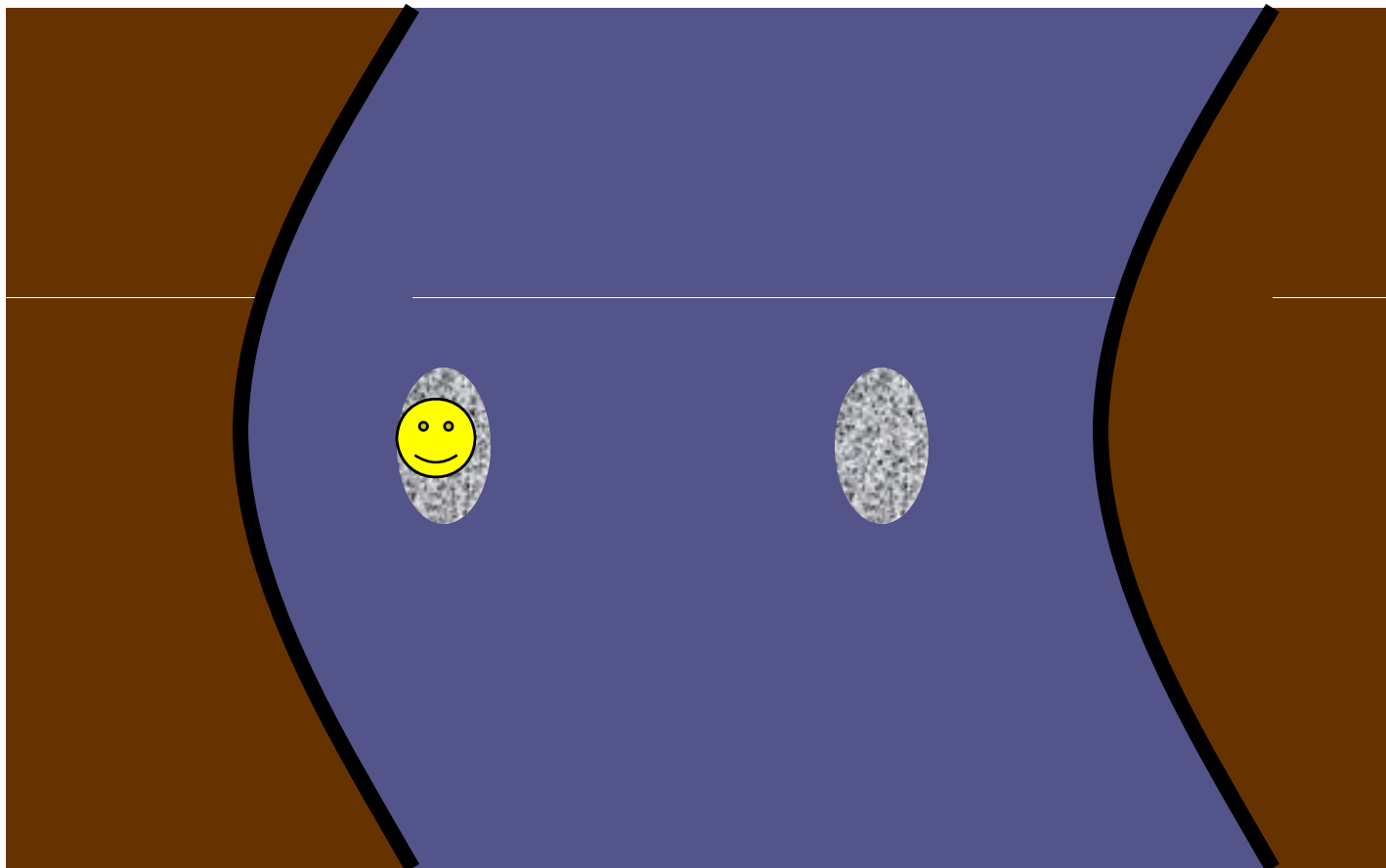
Run



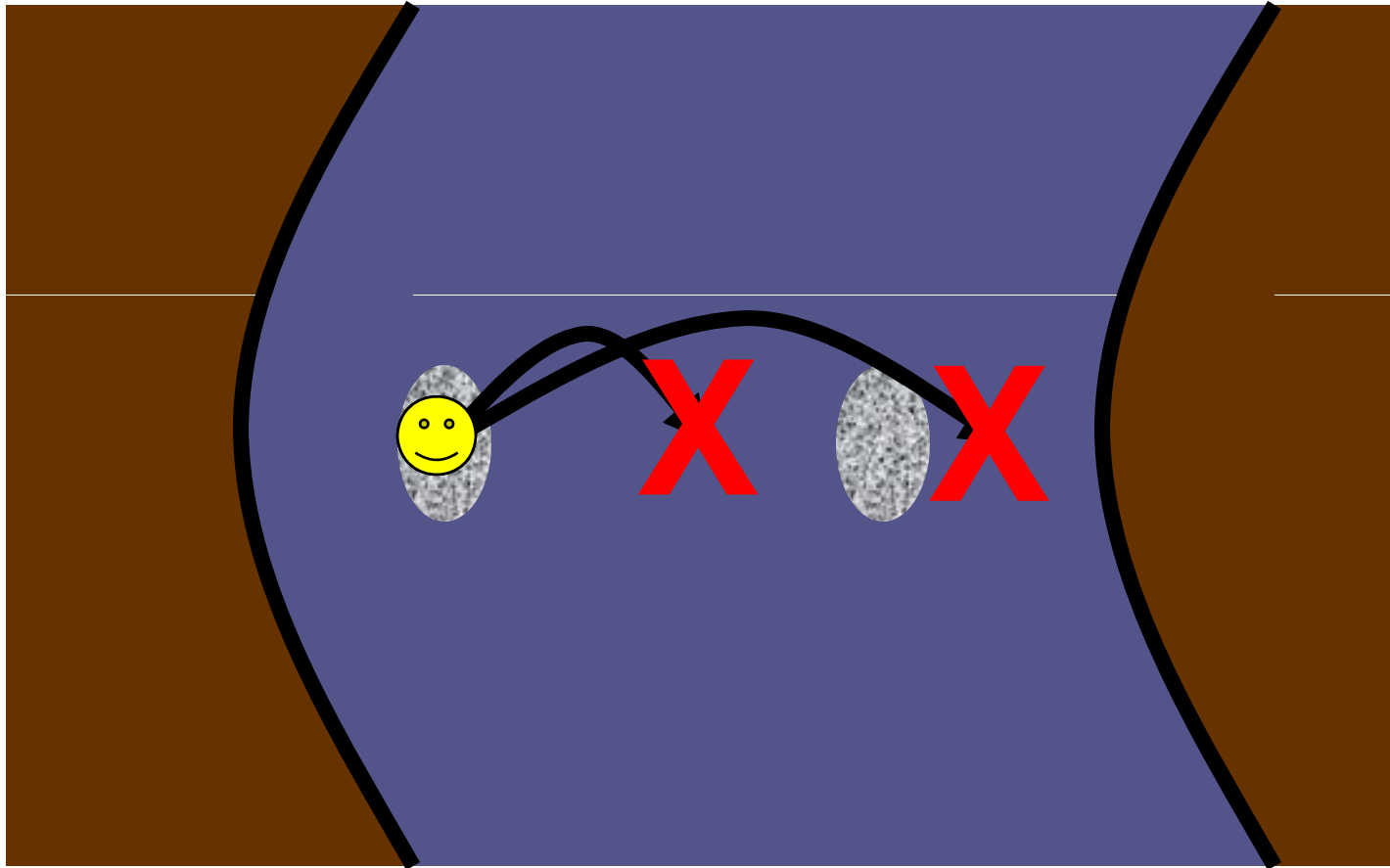
Jump



Jumping Puzzle

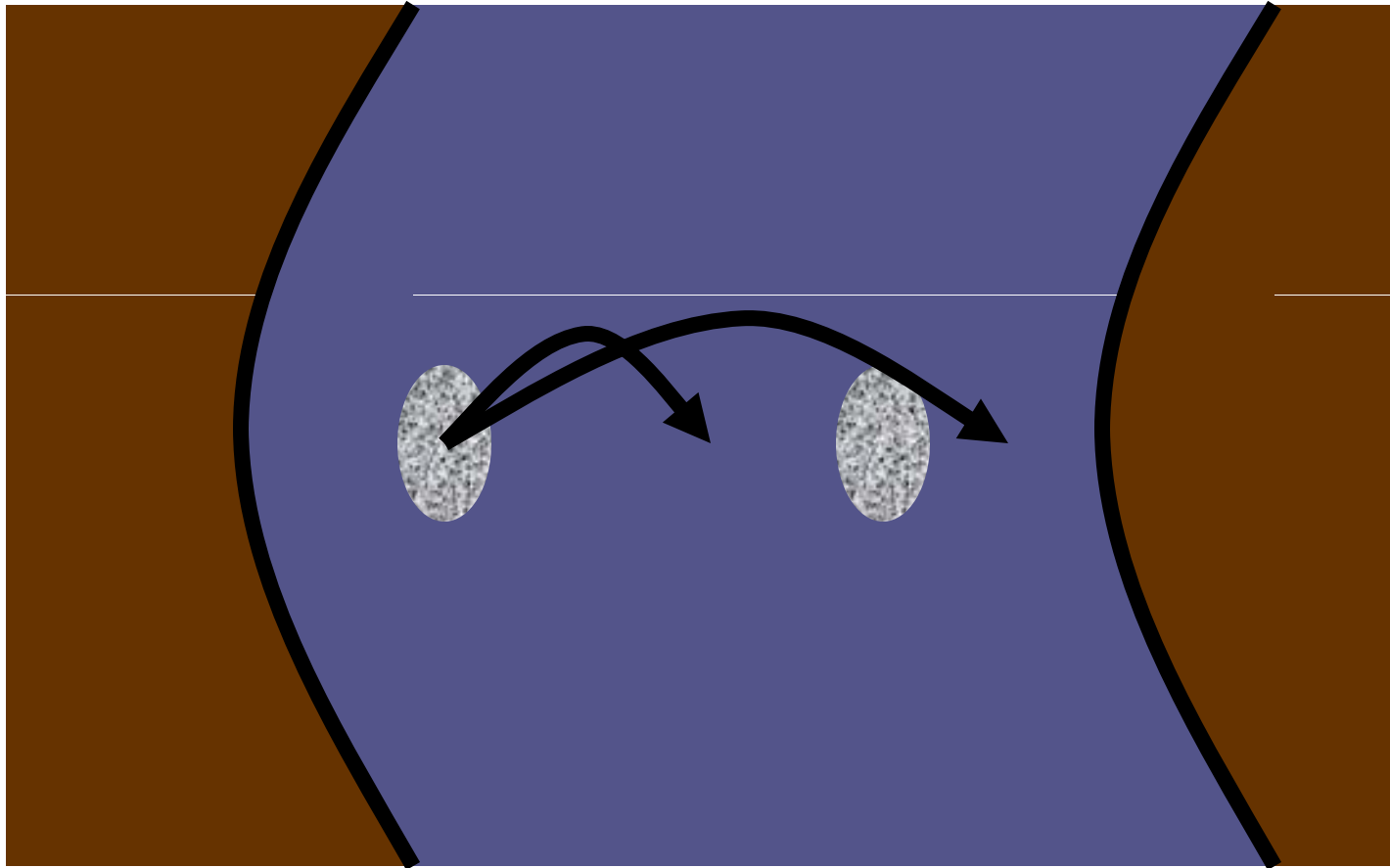


Jumping Puzzle



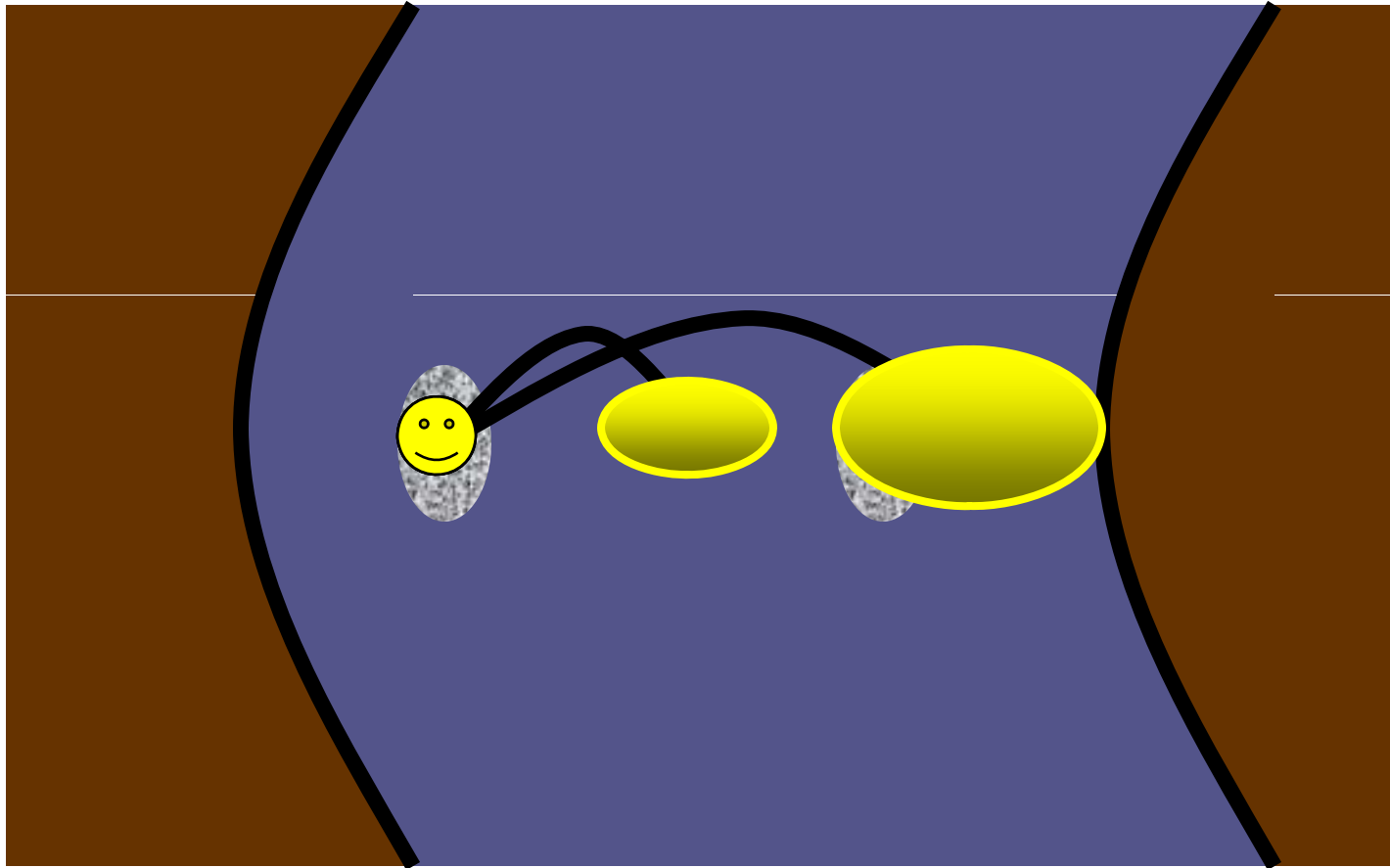
Available source motions can't perform task

Motion Graph Problems



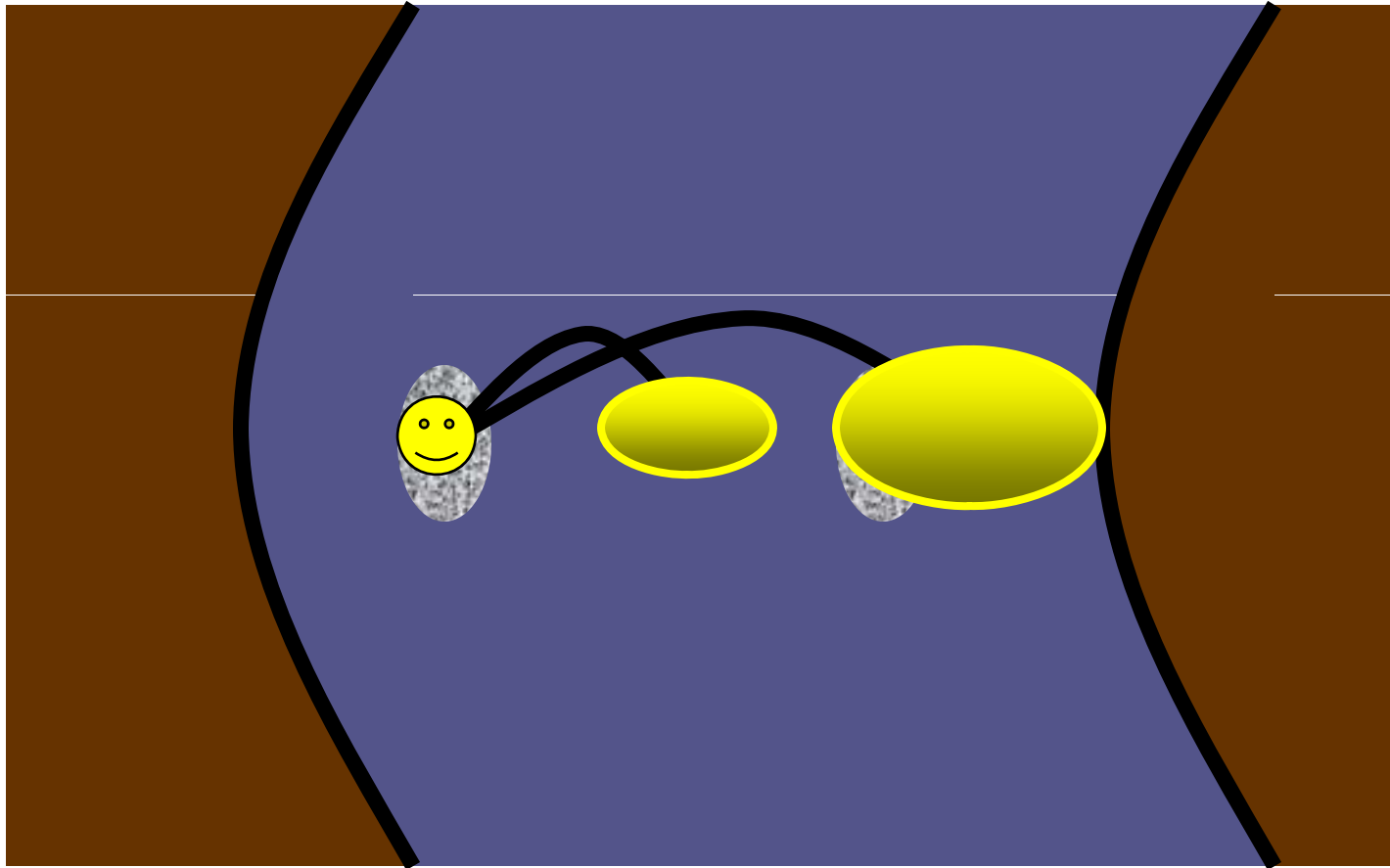
Lack of **motion capability**

Motion Warping



Warp the motion to change jump's landing point

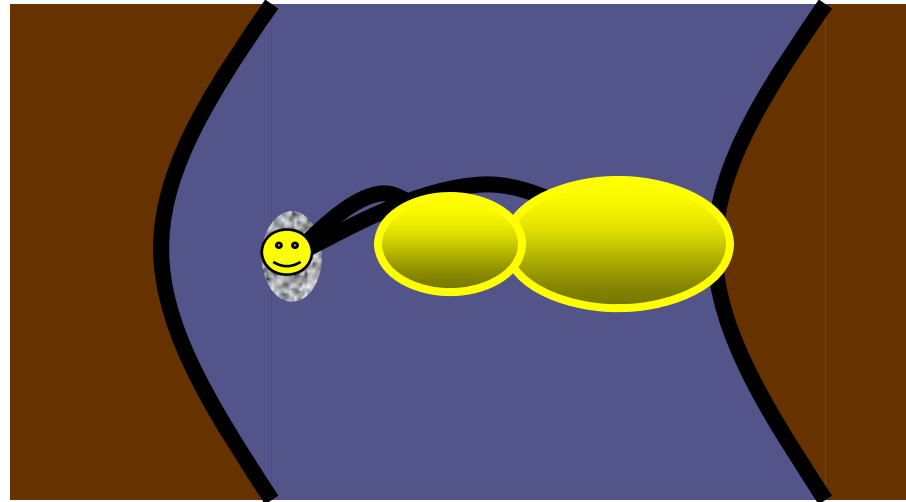
Motion Quality



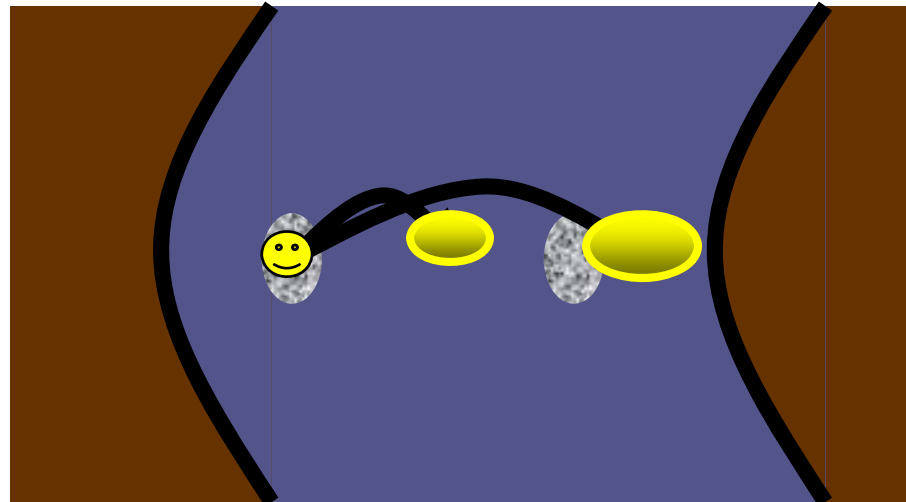
Too much motion editing degrades quality

Capability vs. Quality Tradeoff

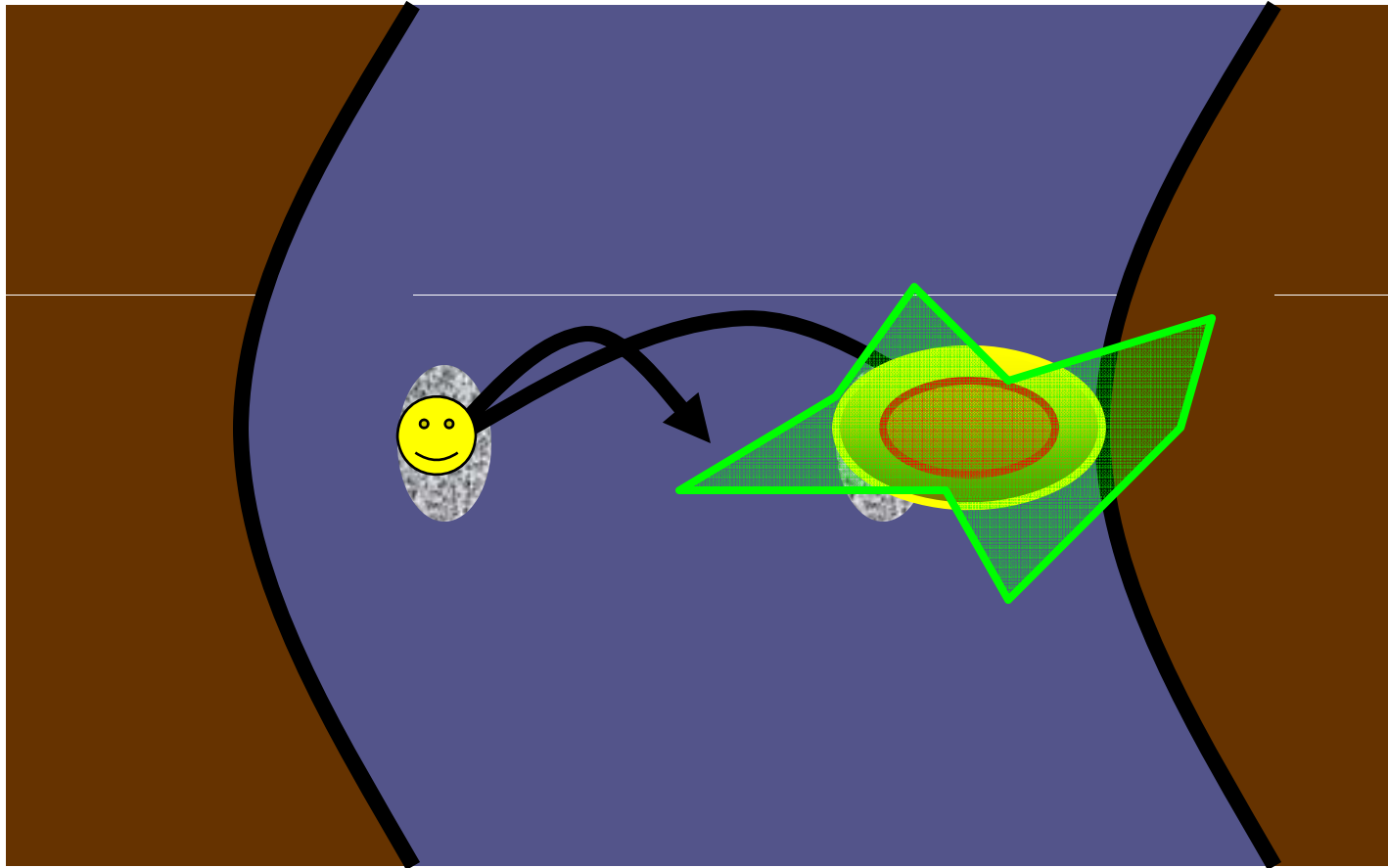
- Low motion
quality okay



- High motion
quality required



What is quality motion?



How much editing is acceptable?

Motion Quality

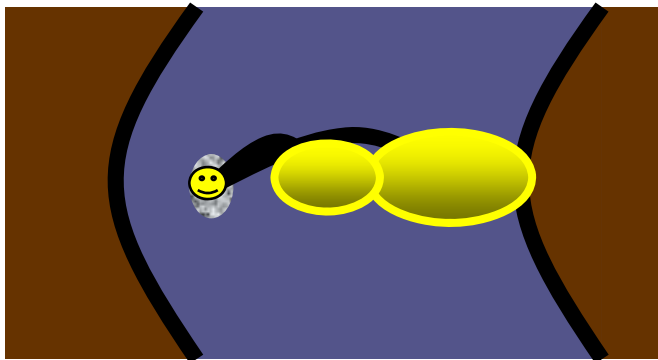
- Another aesthetic problem
 - Cannot be computed from the motion data alone
 - Collect data from people, fit a model

Motion Quality

- Another aesthetic problem
 - Cannot be computed from the motion data alone
 - Collect data from people, fit a model
- A model of human perception of motion?
 - Tall order, new effort
 - Guided by success in other areas (rendering)

Approach: Measure Perceived Error

- How noticeable are these errors?
 - How do we measure perceived error?
- How much error is acceptable?
 - How can we jump that river?



Other Data Structures

- Fat Graphs
 - Edges are *sets* of clips between the poses
 - Using an edge means blending the set
 - Powerful method for creating motion families, but no change to reactivity

Other Data Structures

- Fat Graphs
- Move Trees
 - Hand-designed motion graphs
 - Tradeoff between time/skill and reactivity
 - May be infeasible for getting fast enough

Other Data Structures

- Fat Graphs
- Move Trees
- Blend Trees
 - Hand-designed network of clips that are blended together to create current motion
 - Change behaviour by changing blend weight

Blend Trees

- Build tree based on current behaviour
 - e.g., jogging (=run+walk) while throwing and falling (=standing+stumbling+lying)
- Advance each animation
- Update weights, position of each source
 - e.g., advance falling state to lower
- Blend all sources together
- Add IK, procedural animation, etc.

Blend Tree Pros

- Automatic quality/reactivity tradeoff
 - Heavily used in games industry
 - Only as good as its inputs
- Other tools fit well into pipeline
 - IK, procedural, physics-based, ...
- Leverages skill of animators

Blend Tree Cons

- Requires skill of animators
 - Not all of us have them handy
 - Rapidly becomes complex, time-consuming
 - e.g., overwhelmed the MechWarrior team
- Produces blended motion
 - May destroy desirable details of source motion
 - e.g., characteristic mannerisms of star player
- Resource-intensive
 - More approximations means more damage

Meaning?

- Where does that leave the character?
- Pretty but dumb
 - Basic motion graphs give great motion but poor control
- Better but vapourware
 - Optimized and hybrid motion graphs are still research, and are harder to implement
- Controllable but at risk
 - Blend trees risk degrading the animation even away from transitions
 - Current best bet for highly interactive

More Information

- Motion Graphs
 - “Motion Graphs”, Kovar and Gleicher, 2002
 - “Evaluating Data-Driven...”, Reitsma and Pollard, 2007
 - Steve, Nithin, and I
- Perception
 - “Perceptual Metrics...”, Reitsma and Pollard, 2003
 - “Evaluating the Visual...”, O’Sullivan et al. 2003
 - “...Quantifying Natural Human Motion”, Ren et al., 2005
- Blend Trees
 - www.gamasutra.com/features/20030704/edsall_03.shtml