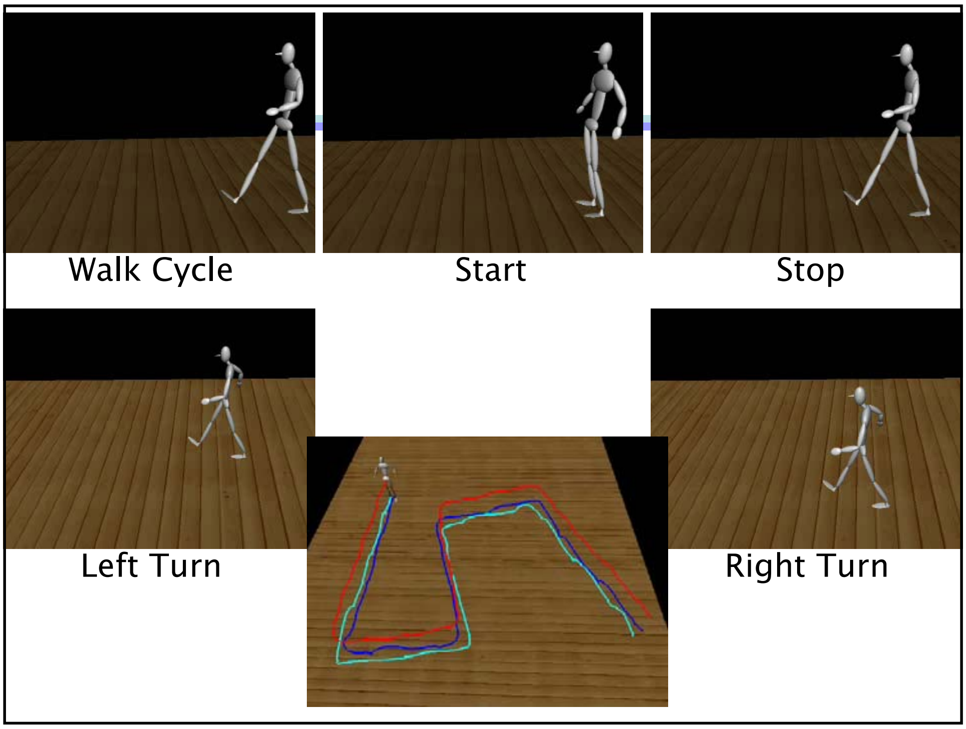
Motion Graph

Jehee Lee
Seoul National University

Motion Database

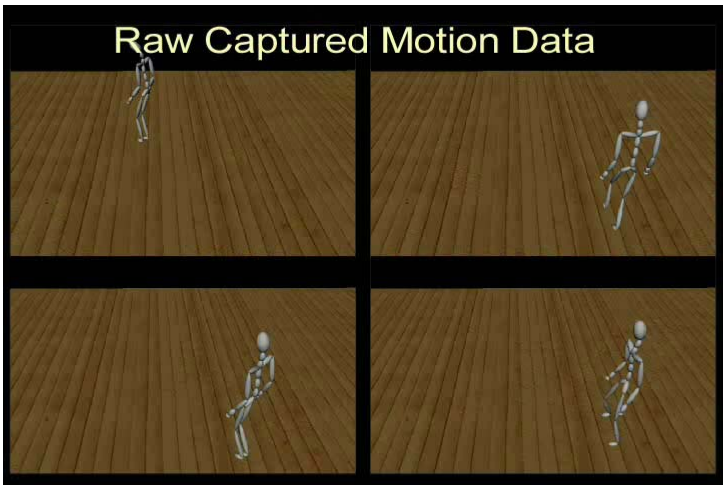
- **In video games**
  – Many short, carefully planned, labeled motion clips
  – Manual processing



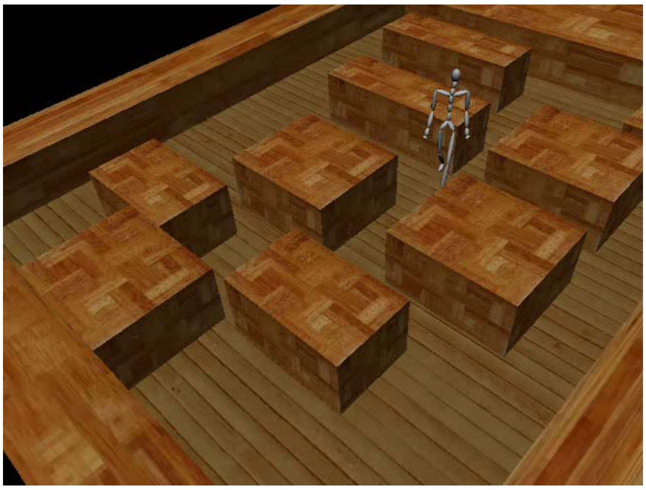
Motion Database

- **Motion graphs**
  - – Extended, unlabelled sequences
  - – Automatic processing

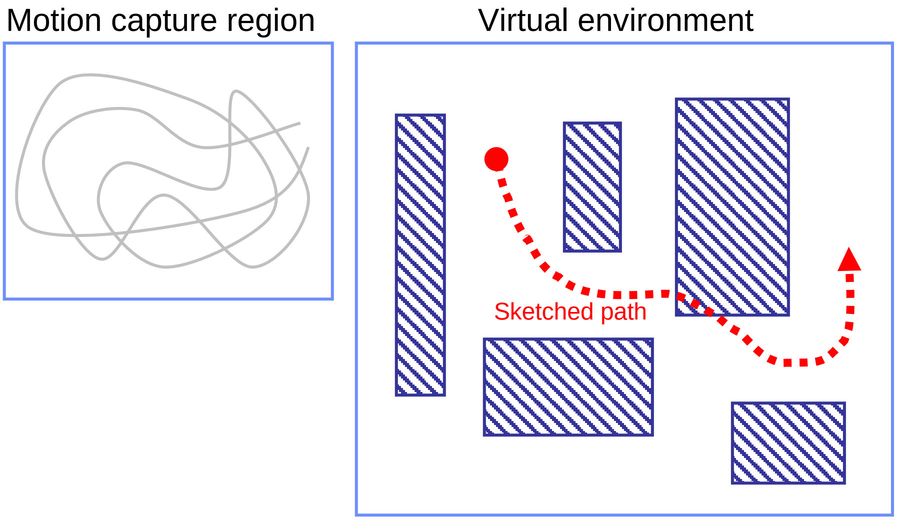
Motion Data Acquisition



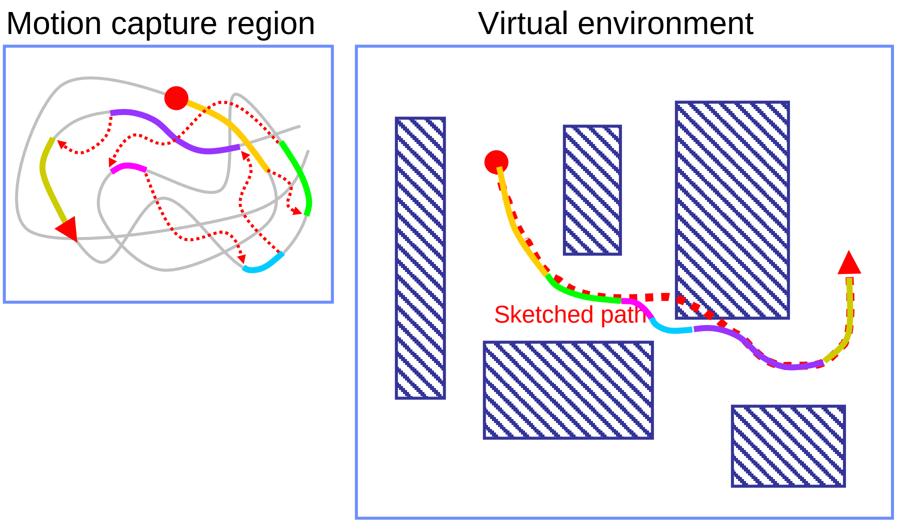
Maze – Sketch Interface



Re-sequence motion frames



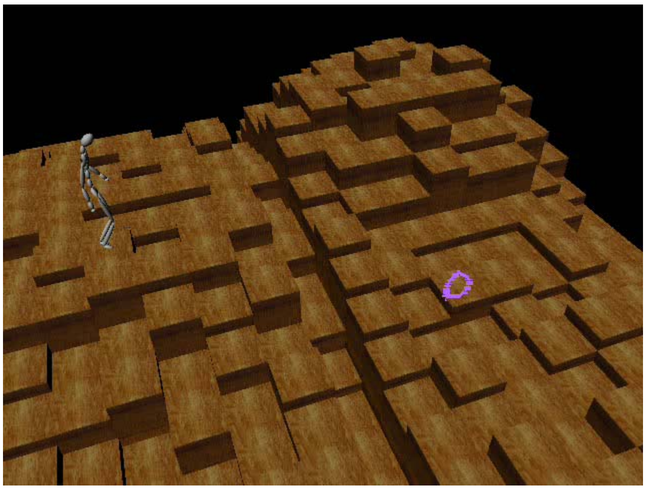
Re-sequence motion frames



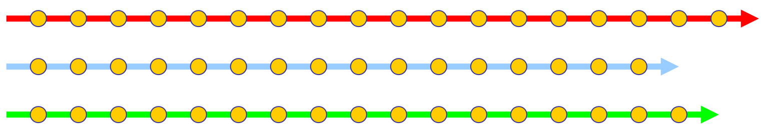
Data Acquisition

- "Poles and Holes" rough terrain

Terrain Navigation

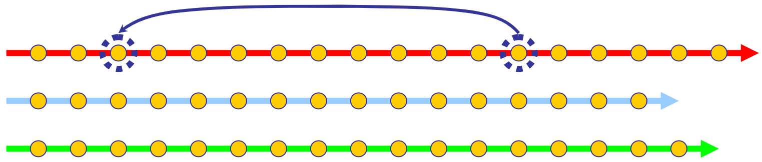


Unstructured Input Data



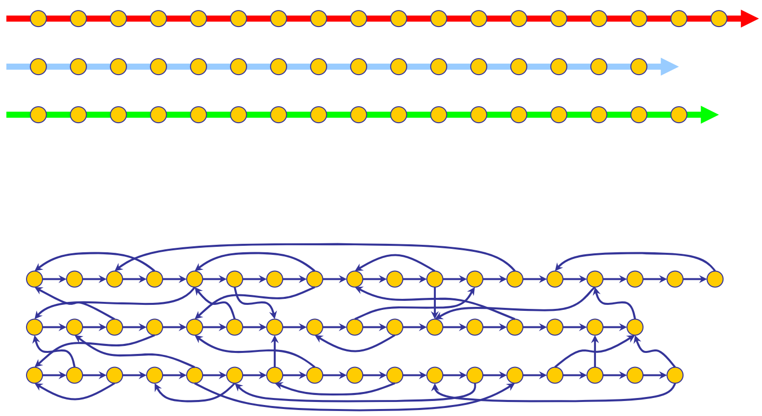
- A number of motion clips
  - Each clip contains many frames
  - Each frame represents a pose

Unstructured Input Data



- Connecting transitions
  - Between similar frames

Unstructured Input Data

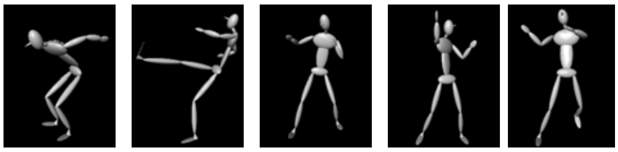


Pose Distance

$$D(i,j) = d(p_i, p_j) + \alpha\, d(v_i, v_j)$$

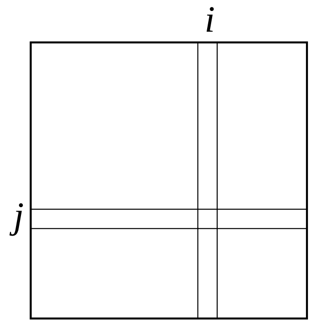
Weighted differences
of joint angles (or positions)

Weighted differences
of joint velocities



Pruning Transitions

- **Reduce storage space**
  - – O(n^2) will be prohibitive
- **Better quality**
  - – Pruning “bad” transitions
- **Efficient search**
  - – Sparse graph



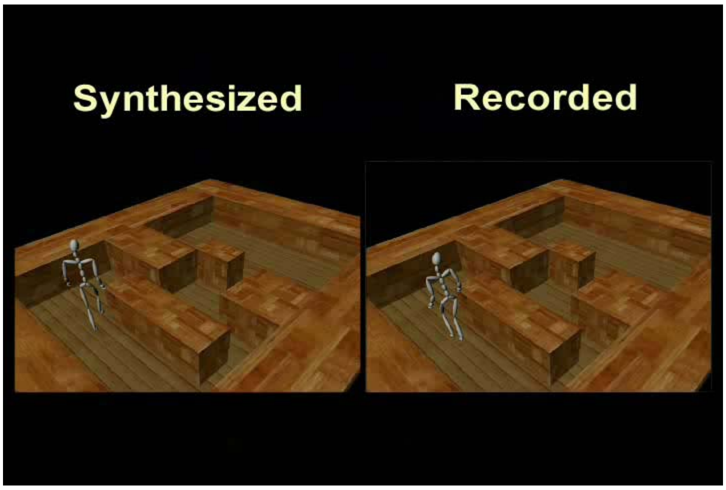
Pruning Rules

- **Contact state**: Avoid transition to dissimilar contact state
- **Likelihood**: User-specified threshold
- **Similarity**: Local maxima
- **Avoid dead-ends**: Strongly connected components

Comparison to Real Motion

- Environment with Physical Obstacles

Comparison to Real Motion



Data-Driven Keyframing

- **Inbetweening**
  - Pose
  - Location (or continuous path)
  - Pose and Time
  - Pose and Location
  - Pose and Time and Location

Data-Driven Keyframing

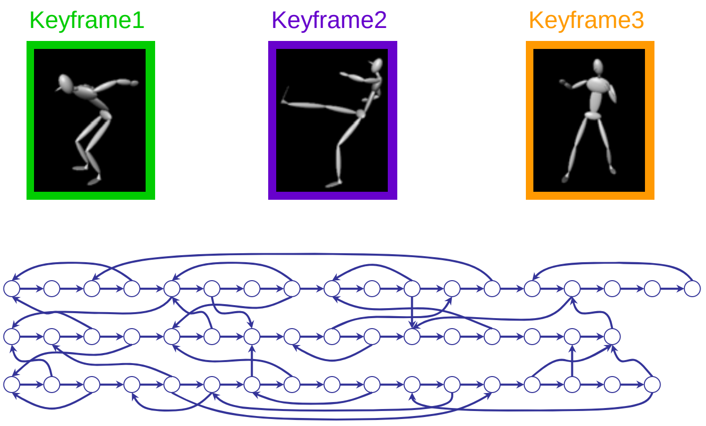
- **Criterion**
  - Pose Accuracy
    - In general, there is no exact match
    - Motion editing techniques can be used to achieve exact interpolation
  - Minimum time
    - # of frames traversed through motion graph
  - Minimum distance
    - Distance traversed by the center of gravity

Search Techniques

- Graph traversal
  - Pose-to-pose inbetweening
- Branch-and-Bound
  - A*-algorithm
  - Limited horizon on-line algorithm
- Dynamic programming
  - Pose-Time space
  - Pose-Location space

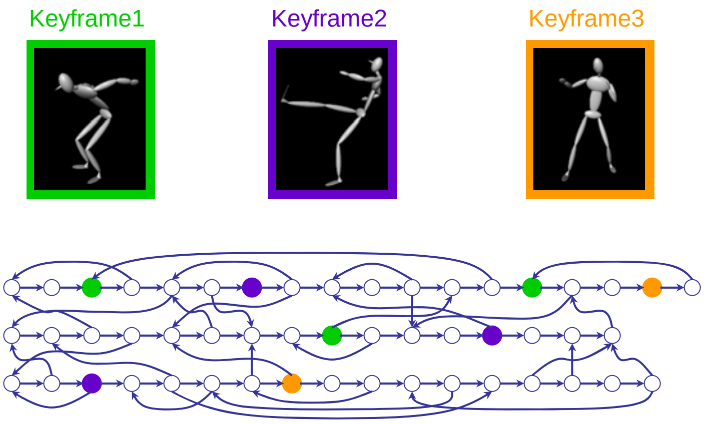
Graph Traversal

- Pose-to-pose inbetweening

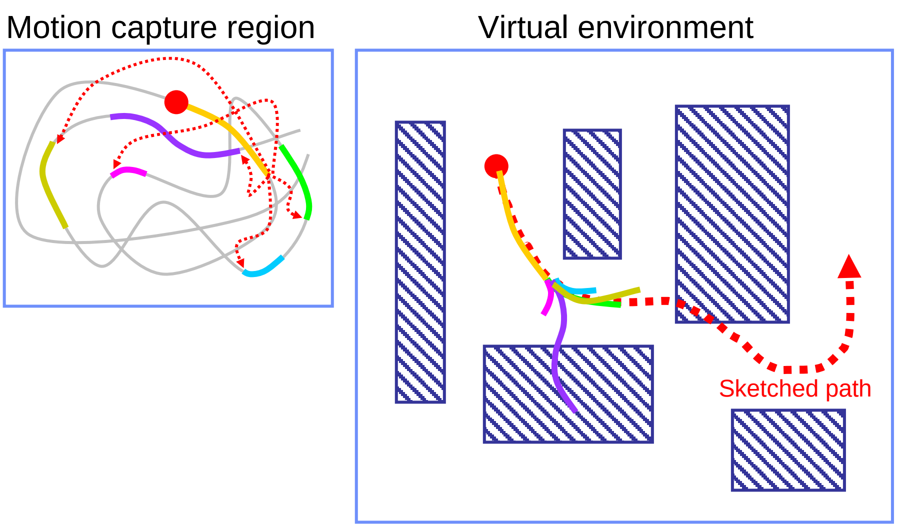


Graph Traversal

- Pose-to-pose inbetweening



Branch-and-Bound Search



Branch-and-Bound Search

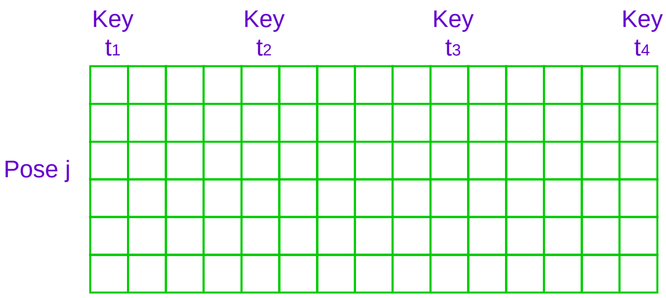
- A*-algorithm
  - Optimal search techniques
  - Infinite horizon
  - Competitive with discrete state space
  - Slow and memory intensive in continuous domain
  - pathFit.avi (Kovar et al 2002)

Branch-and-Bound Search

- On-line search
  - Non-optimal
  - Limited horizon
  - Much faster than A* algorithm
  - Better suited for on-line applications

Dynamic Programming

- Pose and time



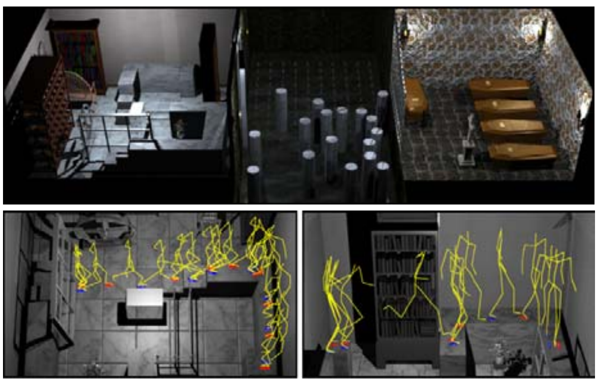
$$D(j,t)=\min_{\hat{j}}\left(D(\hat{j},t-1)+T(\hat{j},j)\right)+K_t(j,k)$$

Transition cost ($T(\hat{j},j)$)

Keyframe match cost ($K_t(j,k)$)

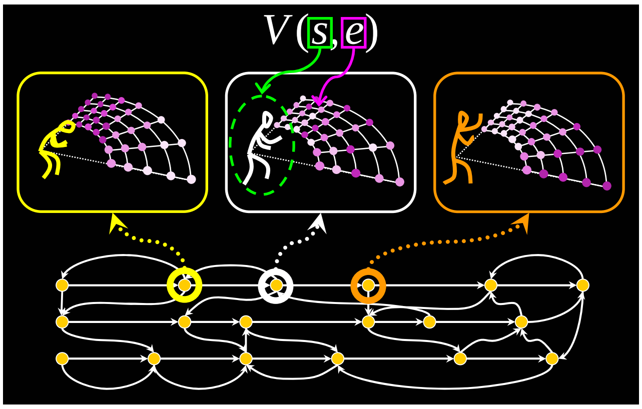
Dynamic Programming

- Pose and (fixed) location
  - Randomly sample locations
  - min_full.avi (Choi, Lee, Shin 2003)



Dynamic Programming

- Pose and (relative) location
  - Precompute and tabulate expected future rewards
  - SCA_MocapBoxer.mov (Lee and Lee 2004)



Data-Driven Keyframing Summary

- **Search techniques for inbetweening**
  - Pose : graph traversal
  - Location : BB or DP
  - Pose and Time : DP
  - Pose and Location : BB or DP
  - Pose and Time and Location : BB

(BB : Branch-and-Bound, DP : Dynamic Programming)